1985

Users guide to approximate reasoning and the REVEAL software system.

Verhagen, Douglas Wayne.

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

USERS GUIDE
TO APPROXIMATE REASONING AND
THE REVEAL SOFTWARE SYSTEM

by

Douglas Wayne Verhagen

March 1985

Thesis Advisor: Paul S. Fischbeck

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)*: USERS GUIDE TO APPROXIMATE REASONING AND THE REVEAL SOFTWARE SYSTEM	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis, March 1985
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Douglas Wayne Verhagen	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943	12. REPORT DATE: March 1985
	13. NUMBER OF PAGES: 140
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release: distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Approximate Reasoning
Fuzzy Sets
Linguistic Variables
Fuzzy Set Theory

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This thesis introduces the reader to some of the principles of fuzzy set theory and approximate reasoning. Sample applications are discussed with possible military situations outlined. The REVEAL software package is reviewed and its relationship with fuzzy set theory and approximate reasoning discussed. This thesis is designed to be used as a tutorial for REVEAL.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

Users Guide
to
Approximate Reasoning And The REVEAL Software System

by

Douglas W. Verhagen
Lieutenant, United States Navy
B.S., University of Wisconsin, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
March 1985

ABSTRACT

This thesis introduces the reader to some of the principles of fuzzy set theory and approximate reasoning. Sample applications are discussed with possible military situations outlined. The REVEAL software package is reviewed and its relationship with fuzzy set theory and approximate reasoning discussed. This thesis is designed to be used as a tutorial for REVEAL.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENT

The author would like to express his most sincere appreciation to LCDR Paul Fischbeck and Professor Norm Lyons. They provided direction, encouragement, and patience throughout the thesis development process.

Finally, he would like to thank his wife, Cathy, and sons, Mike and Joe, for through their patience, understanding and support, made graduate school enjoyable.

I. INTRODUCTION

A. DISCUSSION

The efficient use of computer systems has, over the years, made for effective decision making by providing managers with accurate information in a timely manner. Various models and modelling languages have sprung to meet this demand. They have not been entirely successful. Computers are extremely rigid and precise information systems while people understand and operate on vague natural language concepts. These characteristics between man and machine severely limit a computer's ability to abstract useful information and generalize fundamental concepts. A recent attempt at trying to bridge this man-machine gap in the modelling process finds its roots in fuzzy set theory developed by Lotfi Zadeh and its extension of Approximate Reasoning. Fuzzy thinking attempts to make possible the solution of problems that are too complex for precise analysis and that more closely resemble real world situations. It steps beyond the limits imposed by precise mathematical methods in systems analysis. Fuzzy set theory works well in a problem atmosphere where either some variables are ill defined or the relationship between many variables is ill defined.

Knowledge bases in most modelling situations are imprecise, incomplete, or not totally reliable. Managing this uncertainty is a problem. To deal with this uncertainty in a systematic way fuzzy sets and approximate reasoning are used. This allows such quantifiers as few, many, high, large, tall, etc. to be used and used effectively. Rarely are real world variables totally true

or totally false. More often the variable takes on a value somewhere between. As an example, we might have the real world situation in inventory control of "if demand is high then reevaluate the reorder point". What is considered to be high demand? Current programming languages allow for this quantitative analysis, and we could set the high demand at 50 units/quarter. By doing so, however, we are treating a demand of 2 units/quarter on a par with a demand of 47 units/quarter. This clearly is not a realistic representation of the situation. The problem needs to be reviewed from a qualitative aspect. Fuzzy sets and approximate reasoning allow us to do this. A basis is provided for a systematic approach to realisticly solve this problem. Fuzzy sets and approximate reasoning have an impact in many areas including psychology, economics, law, medicine, decision analysis, information retrieval, and artificial intelligence.

B. STRUCTURE OF THESIS

This thesis is designed to introduce the reader to the basic concepts of Fuzzy Set Theory and Approximate Reasoning. Chapter Two discusses the basic underlying principles. Practical military applications are discussed in the next two chapters. A simple non-tactical inventory model is discussed in Chapter Three and a tactical model is discussed in Chapter Four. REVEAL is introduced in these two chapters, and discussion centers around its use as a modelling tool for effective decision support. The concepts of fuzzy set theory and approximate reasoning are linked to the computer through REVEAL. Conclusions and observations that resulted from this work and suggested further research are covered in the final chapter. Programs used in the inventory model and tactical model are listed in Appendix A

and Appendix B respectively. A program for an online tutorial is listed as Appendix C.

II. FUZZY SETS AND APPROXIMATE REASONING

A. OVERVIEW

Fuzzy set theory was proposed by Lotfi Zadeh in 1965 as a means of dealing with problems too complex for precise solution. In this chapter we will cover the basics of fuzzy set theory and approximate reasoning to give the reader a general background for use in programming with REVEAL.

B. FUZZY SETS

When we talk about the class or set of countries of the world, the members of this set are clear. China, United States, France, Canada, Sri Lanka, etc. all belong to this set. However, if we talk about the class or set of tall men or big objects or beautiful women, the set membership is unclear or fuzzy. A fuzzy set is a class in which there may be a continuum of membership as in the class of large ships. Such a set is characterized by a membership or characterization function which assigns to each object a grade of membership ranging from 0 (totally false) to 1 (totally true). The transition in this continuum from membership to non-membership is gradual and specifiable unlike a step function with .5 being the crossover point from somewhat true to somewhat false. A ship that has a length of 1000 feet is definitely long and one with a length of 200 feet is definitely not long, but what of the ship that is 600 feet? Such sets are numerous in the real world. Submarines dive deep, jets fly high, cruise missiles fly low, and projectiles hit close. Fuzzy sets play an important role in human cognition and underlie our ability

to make effective decisions with partial information or uncertainty. The notions of inclusion, union, intersection, complement, etc. are extended to these sets, and various properties of these notions, in the context of fuzzy sets, are established. Linguistic as well as numeric variables can be acted on. [Ref. 1]

C. TERMINOLOGY

Fuzzy set theory has its own terminology, some unique and some borrowed from other disciplines. A capital letter, one towards the end of the alphabet, is used to designate a space or set of objects. As an example, X = Set of all ships, or Y = Set of all aircraft. A general element of the set, an aircraft carrier or P-3, is denoted with the lower case letter of the set. A capital letter, one towards the beginning of the alphabet, denotes a fuzzy set (class) in the set of objects, ie. A = set of large ships or B = set of fast aircraft. A fuzzy set is characterized by a membership (characterization) function, fA(x), which associates each point in X with a real number in the interval [0,1.0] with 0 being totally false and 1 being totally true and .5 being the crossover point. [Ref. 2]

Membership functions can be determined in a variety of ways. See Figure 2.1. Analyzing the set of ships further, we will let A = set of all long ships, and the database from Table I will be used[1] [Ref. 3].

One can then give a precise although subjective characterization of A by specifying fA(x) as either an enumeration, table, algorithm, or function. Membership in a fuzzy set (as in non-fuzzy sets) is specified by mapping from the universe, all ships, to the set in question, long

---

[1] Displacements cited are unloaded for ships and surfaced for submarines.

1) Enumeration: A formula is used to determine membership. For example-- If length< 100 then grade of membership in short is 1.0.

2) Table: A table is used to determine membership. See Table II for an example.

3) Algorithm: Recursive calls within and algorithm. For example-- normal is not tall and not short.

4) Function: A function determines membership. For example-- very A = $fA^2(x)$.

Figure 2.1 Membership Function Methods

TABLE I

Ships and Their Lengths and Displacements

Ship	Hull No.	Length (ft)	Displacement (1000 tons)
Ohio	SSBN 726	560	16600
Sam Rayburn	SSBN 635	425	7250
Omaha	SSN 692	360	6000
Narwhal	SSN 671	314	4450
Tullibee	SSN 597	273	2317
Bonefish	SS 582	219	2145
Carl Vinson	CVN 70	1092	72798
Iowa	BB 61	887	45000
Virginia	CGN 38	585	8623
Albany	CG 10	674	13700
Preble	DDG 46	512	4150
Garcia	FF 1040	414	2620
Pegasus	PHM 1	140	239
Iarawa	LHA 1	820	39300
Manitowoc	LST 1180	522	8450
Frank Cable	AS 40	643	13000

ships. To determine the membership function for our example we will use Table II, a completely subjective opinion of this author.

TABLE II

Grade of Membership of Length to Ships

Length (ft)	Grade of Membership long	medium	short
100	0	0	1.0
200	0	0	1.0
300	0	.2	.5
400	0	.5	.2
500	0	1.0	0
600	.1	.5	0
700	.2	.2	0
800	.5	0	0
900	.7	0	0
1000	1.0	0	0
1100	1.0	0	0

The elements of a fuzzy set can then be listed as ordered pairs in the form {fA(x),x}, where fA(x) is the grade of membership of x in A. Here, the first number is the grade of membership of lengths in long ships.[2]

Therefore:

long ships (A) = {(.1,Ohio),(1.0, Vinson),(.7,Iowa),
(.1,Virginia),(.2,Albany),
(.7,Tarawa),(.2,Cable)}

Ships with a degree of membership of 0 are omitted for brevity. It is important to note that for non-fuzzy elements, the grade of membership can only be 0 or 1.

D. FUZZINESS VS. STATISTICS

At this point we diverge slightly to point out an important distinction. Fuzziness is different from probability statistics despite any perceived similarities. The uncertainty of a coin toss resulting in a heads has a

---

[2]Lengths rounded to nearest hundred before entry into table II

certain probability associated with it. There is no vagueness on the outcome--It will be either a head or a tail. There is only a lack of knowledge concerning this outcome. Once this knowledge becomes available the state of affairs is completely determined. However, fuzziness is not so precise. Take the fuzzy concept of tall. Unlike the coin toss no matter how close we measure, scrutinize, or examine the concept of tall will apply more to some buildings than to others. There is no precision in this boundary and there is no amount of information that will make the boundary between tall buildings and not tall buildings clear. The set of tall buildings is fuzzy. [Ref. 4]

E. FUZZY SET CHARACTERISTICS AND PROPERTIES

Once mapping is specified the fuzzy set can be analyzed and used as a linguistic variable in fuzzy inferences and fuzzy algorithms. The set can be modified by operations such as negation, union, DeMorgan's law, algebraic sum, etc. The relationship may be extended by having the grade of membership itself being a fuzzy set. For example: if X = {Ray, Vinson, Hancock, Petril}, and A is the fuzzy set of fast ships, the following may apply: fast ships = {(medium, Ray), (high, Vinson), (high, Hancock), (low, Petril). Note how the grades of membership, (high, medium, and low) are fuzzy sets themselves.

The following properties apply to fuzzy sets [Ref. 5].

1. Equality in Fuzzy Sets

Fuzzy set A is equal to fuzzy set B if and only if $fA(x) = fB(x)$ for all x in X. For example if X = {Ohio,

Omaha, Ray} then if:

long ships (A) = {(.8,Ohio),(.3,Omaha),(.1,Narwhal)}

long subs (B) = {(.8,Ohio),(.3,Omaha),(.1,Narwhal)}

then A is equal to B.

2. Complements of Fuzzy Sets

The complement of fuzzy set A is A' and is determined by the equation fA' = 1 - fA. For example if:

long ships (A) = {(1.0,Vinson),(.7,Iowa),(.2,Albany)}

then

not long ships (A')= {(0.0,Vinson),(.3,Iowa),(.8,Albany)}

3. Containment (Subset)

Set A is contained in set B if and only if fA <= fB. For example if:

long ships (A) = {(1.0,Vinson),(.1,Ohio),(.7,Iowa)}

and

long submarines (B) = {(.1,Ohio)}

then B is a subset of A.

4. Union

The union of two fuzzy sets results in a third fuzzy set. The union of fuzzy set A with fuzzy set B is the smallest fuzzy set containing both A and B. fC(x) = MAX [fA(x), fB(x)] or fC = fA V fB. For example if:

long ships (A) = {(.7,Iowa),(.1,Ohio),(.2,Albany)}

and

medium ships (B) = {(0.0,Iowa),(.5,Ohio),(.2,Albany)}

then

long or medium ships (C) =

{(.7,Iowa),(.5,Ohio),(.2,Albany)}

5. Intersection

The intersection of fuzzy set A with fuzzy set B results in a third fuzzy set, C, such that fuzzy set C is the largest fuzzy set containing both A and B. $fC(x) = MIN [fA(x), fB(x)]$ or $fC = fA \wedge fB$. For example if:

long ships (A) = {(.7,Iowa),(.1,Ohio),(.2,Albany)}

and

medium ships (B) = {(0.0,Iowa),(.5,Ohio),(.2,Albany)}

then

long and medium ships (C) =

{(0.0,Iowa),(.1,Ohio),(.2,Albany)}

6. Other Properties

Many other arithmetic properties that apply in set theory also apply to fuzzy sets. Assuming A, B, and C are fuzzy sets then the following properties apply:

Commutive	A and B = B and A
Associative	(A and B) and C = A and (B and C)
Distributive	A and (B or C) = (A and B) or (A and C)
Idempotent	A and A = A
Adsorptive	A and (A or B) = A

These properties likewise hold for 'or'.

7. Linguistic Operators

It has been shown that various linguistic operators (ie. very, rather, slightly, etc.) can enhance the grade of membership and become a part of fuzzy logic by being considered as additional operators on linguistic variables. Adverbs and adjectives can intensify or lessen a grade of membership. Taking the adverb, 'very', as an example, Zadeh [Ref. 6] has shown that 'very A' can be of the form:

$$\text{very A} = \{fA^2(x), x\}$$

This would show that if, for instance, x had a grade of membership of .8 in A, the set of long ships, the membership in very A, the set of very long ships, would be

.64. Note that very long ships still lies within the interval 0 to 1.0. Other possible operators include quite, about, and not. It must be realized that this calculation is subjective based on the viewpoint of the user.

8. Universe of Discourse

The Universe of Discourse is a collection of objects that is rich enough to make it possible to identify any concept, within a specified set of concepts, with a fuzzy subset [Ref. 7]. The set of ships is one such set of concepts. The Universe of Discourse could be the fuzzy subsets: long ships, medium length ships and short ships. Another possible Universe of Discourse could be the set of all fast ships, slow ships and medium speed ships. See Figure 2.2 for a graphic example.

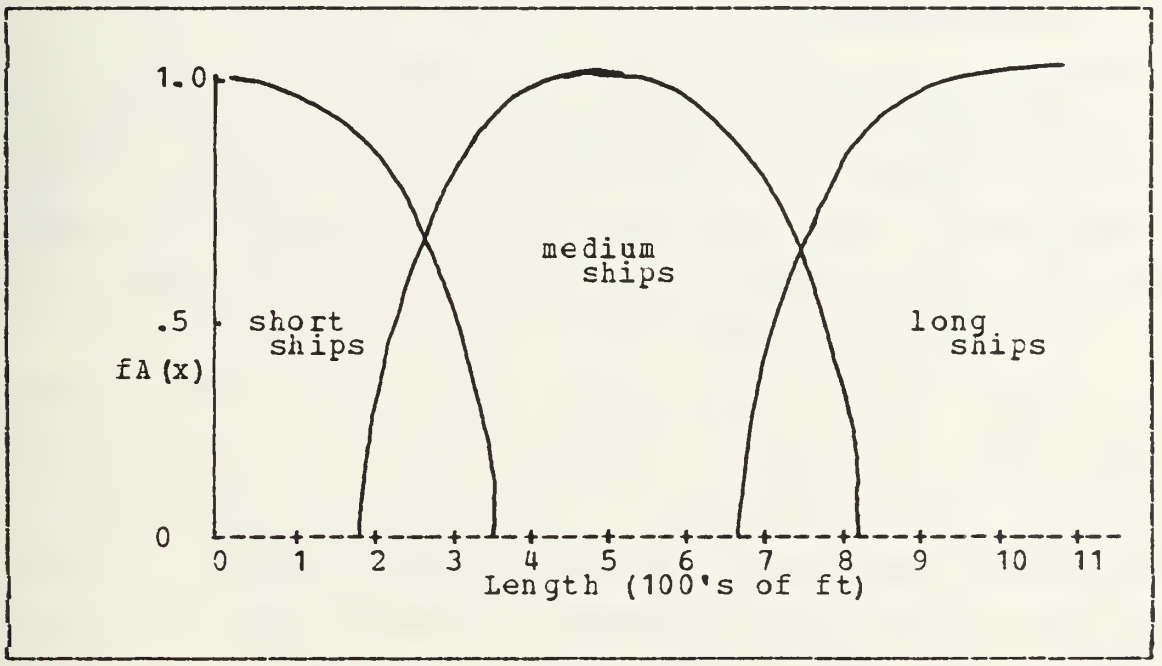


Figure 2.2 Universe of Discourse for Ships

F. SUBJECTIVITY AND FUZZINESS

Fuzzy set theory, as you may have noticed, contains a great deal of subjectivity. This makes the modelling of real world situations flexible. After all, my idea of long ships may differ markedly from the CNO's idea of a long ship. Psychological experiments were performed in an attempt to determine if there was some standard or common factor underlying people's perception of linguistic variables. For example, an attempt was made to determine the validity of Zadeh's definition of 'very'. Although these experiments were not totally conclusive, they did show that natural language concepts, for the most part, are fuzzy in nature and that the theory of fuzzy sets and approximate reasoning are accurate reflections of how people communicate. [Ref. 8]

G. FUZZY ALGORITHMS

Once our vocabulary is set up and fuzzy sets defined, we can then write fuzzy algorithms to aid in the decision making process. Fuzzy algorithms contain fuzzy instructions. This linguistic modelling approach overcomes one difficulty, namely the requirement of numeric precision, in the modelling process.

Many decision support systems employ variations of fuzzy algorithms. Two popular expert systems that are available commercially are MYCIN, a program used to diagnose medical problems, and PROSPECTOR, a program designed to analyze geological formations for mineral deposits [Ref. 9]. These programs contain fuzzy algorithms although fuzzy set theory itself is not used in their problem analysis. Figure 2.3 displays a typical MYCIN rule. Note the fuzzy concepts of 'not known with certainty', 'seriously burned' and 'weakly suggestive'.

```
RULE 0-47
  IF: (1) The site of the culture is blood, and
      (2) The identity of the organism is not
            known with certainty, and
      (3) The stain of the organism is gramneg, and
      (4) The morphology of the organism is rcd, and
      (5) The patient has been seriously burned
  THEN: There is weakly suggestive evidence that
        the identity of the organism is Pseudomonas.
```

Figure 2.3 Sample MYCIN Rule

A fuzzy instruction which is a part of a fuzzy algorithm can be assigned a precise meaning by making use of the concept of the membership function of a fuzzy set. This is subjective in nature and must reflect the context within which the problem is viewed. The complexity of the problem is usually great when modelling a real world situation. Fuzzy logic provides a natural framework for modelling these real world situations where knowledge is imprecise, incomplete or unreliable. Therefore, the REVEAL software package was developed as a tool to aid managers in solving these complex real world problems. We must rely on fuzzy local goals vice precisely specified objectives when dealing with such complex problems and REVEAL helps in doing that.

III. USING REVEAL AND BASIC MODELLING

A. INTRODUCTION

The best way to familiarize oneself with the REVEAL method of Approximate Reasoning is to work through a simple model. In this chapter an inventory model is developed and programmed. The procedures outlined are felt to be the simplest and most logical for beginning programming in REVEAL. The stocking policies for repair parts onboard ships are diverse and directly affect the material readiness of the ship. Figure 3.1 shows a simple inventory model. The item is stocked at a high limit, usually based on demand. As the item is consumed it reaches a low limit at which point it is reordered. The reordered material takes an amount of time to be processed, shipped, and received. This is known as the Order and Shipping Time. Meanwhile, the item is still being consumed. At no time is it desirable for the onhand stock to dip below the safety level. Notice that the faster the Order and Shipping Time the closer the low limit can be to the safety level.

The military criticality of the part also plays a role in the setting of safety levels. Various factors can affect the setting of high limits, low limits, and safety levels. Among them are demand, military essentiality, endurance loads, the order and shipping times, and funding constraints. The Navy calculates these limits based on endurance loads of 30, 45, 60, or 90 days, order and shipping times of 30, 75, or 90 days, and on certain military essentiality codes. Chapter Six of [Ref. 10] contains various tables used in determining stock levels for ships based on these influencing factors.

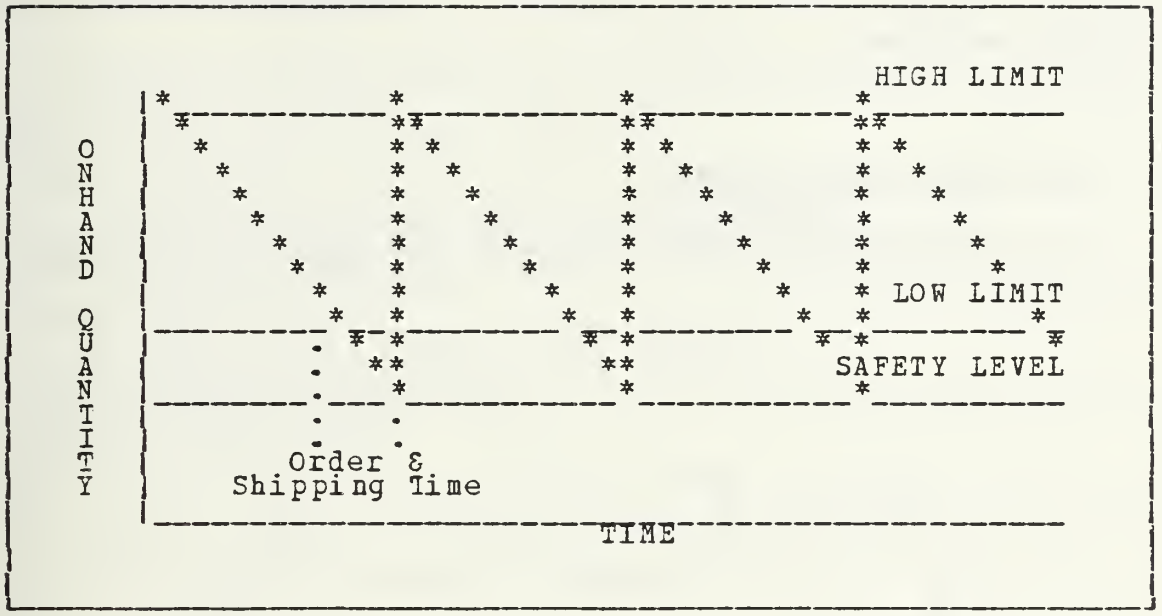


Figure 3.1 Sample Inventory Model

B. APPROXIMATE REASONING AS A MODELING TOOL

The major problem that arises when dealing with these tables can be highlighted in the following situation: A part that has a high demand with an order and shipping time of forty days and a required endurance load of fifty days, must be maintained. It is an essential part, and funding limitations are currently severe for the quarter. Under such a situation entering the tables would be difficult. The numbers can be interpolated, however, the information may be incomplete. This situation lends itself well to modelling with REVEAL. REVEAL is a decision support system that can provide managers more complete information using common English input. Fuzzy sets such as 'high/low demand', 'severe funding limitations', 'fast/slow order and shipping times', and 'short/long endurance loads' can be accurately modelled. A simple inventory model will be developed using some of these fuzzy sets. This model is meant more to introduce the

user to REVEAL than to accurately reflect the Navy's stocking policies.

C. THE REVEAL ENVIRONMENT

The REVEAL environment can best be shown by examining Figure 3.2.

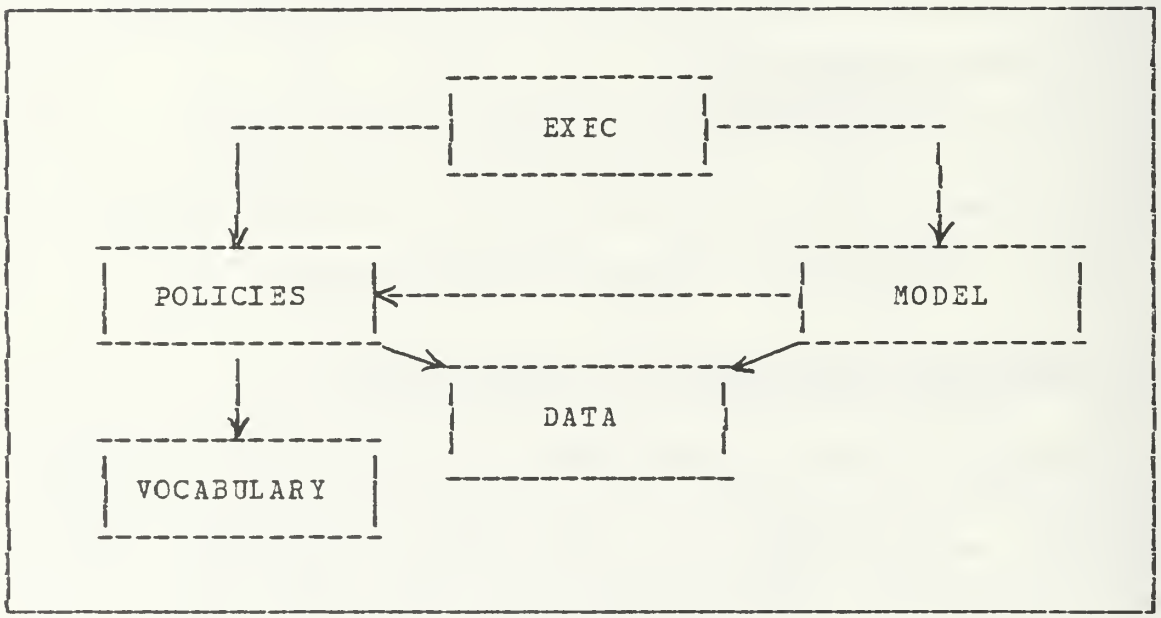


Figure 3.2 The REVEAL Environment

The exec or command executes the model. The model has a certain context or data associated with it. A set of policies is also associated with a particular model. In order for the policies to execute properly, a vocabulary must be created and installed. All variables used in policies must also be defined in the data. Once these conditions are met the model can be run properly.

D. THE USER CONTEXT

Logging onto the computer and entering REVEAL puts the user into a context that contains:

Data Matrices
1) A two dimensional matrix consisting of 1000 series (rows) and 200 terms (columns).
2) A one dimensional matrix of 1000 constants.
3) A one dimensional matrix of 1000 character string variables.

Data Dictionary
1) Series (row) identifiers (names).
2) Term (column) identifiers (headings).
3) Constant identifiers.

Control Parameters
Logic (User Program)

The current user context can be saved by using the 'STORE <filename>' command. A stored context can be retrieved by using the 'LOAD <filename>' command.

E. ENTERING CONTEXT INFORMATION

This sample inventory model will be run for ten items utilizing demand, order and shipping time, endurance load, and military criticality to determine the high limit, low limit, and safety level. Each individual part will be listed as a term (column heading). The command 'HEADING' is entered. REVEAL will then ask for the term number and its name. In this example, term number one is part1, term number two is part2, etc. Cnce terms are entered the series (row) variables can be entered. The command 'NAME' is entered and REVEAL asks for the series number and its name. For this example, series number one is demand, number two is o&s.t, number three is endurance, number four is criticality, number five is high.limit, number six is low.limit, and number seven is safe.level. After terms and series are entered, the constants can be defined by entering the 'CONSTANTS' command. REVEAL will request the constant number, its name and value. If no value is entered the constant can be used as a variable, and the values assigned

and changed by the model program. All constants, names, and headings must be unique consisting of a maximum of twelve characters. Letters, numbers, '.',',' and '&' are allowed. Other characters are allowed, under certain circumstances, and the REVEAL reference manual should be consulted concerning their use. The context can now be stored or used in a modelling situation.

F. DATA ENTRY

Data is assigned to each series by typing 'ENTER <series id> <first term> (last term)'. Once data is assigned to all the terms in the first series it can be assigned to the second series by repeating the procedure. For example, to enter the values for the demand on the first part, you would type 'ENTER s1 t1 t10'. REVEAL will then prompt you for the appropriate data values. Entering data through datasets is also possible. This allows the same model to be run using different data values resulting in greater flexibility.

G. POLICY FORMULATION

The next step in modelling with REVEAL is to enter the vocabulary. However, to do this we must formulate the policies used in the decision making process. Although policies appear to be free form, they conform rigidly to the format 'if <variable> is <fuzzy> then <variable> is <fuzzy>'. The conditional forms the first part of the policy, and the consequent forms the second part. The policies consist of linguistic variables, qualifiers, relations, and noise words. These are all defined in the vocabulary, but first must be identified by reviewing the policies. The following policies will be used: Once the high limit is established it will be increased if endurance is long or if criticality is essential, decreased if endurance

is short or criticality is non-essential, or otherwise left unchanged. If order and shipping time is fast then the low limit can be decreased somewhat; however, if the order and shipping time is slow the low limit should be increased somewhat. Figure 3.3 lists these policies as they will be entered into REVEAL. But first all words must be entered into the vocabulary if they are not already a part of it.

```
POLICY HIGHLIMIT
 If endurance is long or criticality is essential
     then hi.lim.chng is increased
 If endurance is normal then hi.lim.chng is nothing
 If endurance is short or criticality is non-
    essential then hi.lim.chng is decreased

POLICY LOWLIMIT
  If O&S.T is fast then low.lim.chng is somewhat
   decreased
  If O&S.T is slow then low.lim.chng is somewhat
   increased
```

Figure 3.3 REVEAL Policies for the Inventory Model

H. INSTALLING THE VOCABULARY

All words used in policies must be defined in a vocabulary. A vocabulary is first loaded by entering 'VOCABULARY <Vocabulary name>'. The default vocabulary already contains a few qualifiers, hedges, and noise words. Figure 3.4 lists these words. Reviewing the policies shows that qualifiers for 'long', 'normal', 'short', 'increase', 'decrease', 'nothing', 'fast', and 'slow' must be defined. Also, a hedge, 'somewhat', must be defined. All other words are accounted for as variable names, noise words, or relational operators.

QUALIFIERS	NOISE WORDS		HEDGES(value)	Syncnyms
true	my	your	about (-2)	around,near
	their	his	above (-3)	more than
	should	would	below (-4)	less than
	its	to	not (-1)	
	might	must	quite (.5)	
	than	this	very (2)	
	these	those		
	the	an		
	our	her		
	could	a		
	may	that		
	the			

Figure 3.4 REVEAL's Predefined Vocabulary

1. Qualifiers

A qualifier is created by entering 'DEFINE <qualifier name>'. REVEAL will ask for a low limit and a high limit for the dcmain. It will also ask for a function to use in determining the set membership. Figure 3.5 lists the predefined set membership functions. The user may also define his own membership function. A qualifier may also be 'produced' by entering 'PRODUCE <qualifier name>'. This is used when the qualifier to be defined has a relation to scme other qualifier. For example, 'short' may be produced by defining it as 'not long'. Qualifier definition is highly subjective, and slight changes, like defining a qualifier as a grow vice a line, can cause substantial changes in the output. Much care and consideration must be taken when defining qualifiers.

2. Hedges

Hedges act to amplify or decrease the impact of a qualifier. The hedge 'very' was reviewed in Chapter Two. Certain qualifiers are built into REVEAL and should not be

Decline (true, mid, false): declining 'S'-shaped curve.

Grow (false, mid, true): increasing 'S'-shaped curve.

Line (false, true): increasing or decreasing sloped line.

Pi (true, midwidth): Bell-shaped curve

Figure 3.5 REVEAL Set Membership Generation Functions

changed. In the inventory model the hedge 'somewhat' must be defined. To do this the command 'HEDGE <hedge name> <value>' is entered, and here the value of .5 is assigned tc the hedge 'somewhat'. Like qualifiers the values assigned to hedges and the particular hedges used can substantially alter the output.

3. Noise Words

Words that are not used in a logical relation (ie. if, then, or, and) or as a qualifier, hedge, or variable in a REVEAL policy must be defined as a noise word. Noise words are ignored by the compiler when analyzing policy statements. Their sole function is to make policy statements more readable and sound like spoken English. A noise word is created by the command 'NOISE <noiseword>'.

4. Other Functions

Once the vocabulary is created it can be reviewed in a variety of ways. The command 'DICTIONARY' will produce a list of all qualifiers, hedges, and noise words. 'DICTIONARY (qualifier) (hedge) (noise)' will produce a list of the requested words. Qualifiers can be reviewed in three ways. 'DRAW <qualifier name> <qualifier name> ...' will produce a

graph of the requested qualifiers. 'EXAMINE <qualifier name>' will produce a detailed truth function matrix. 'SHOW <qualifier name>' will show a summarized truth function diagram. Qualifiers, hedge and noise words can be deleted by entering 'FORGET <list>' where <list> is a list of words to be deleted.

I. POLICY ENTRY

Once a vocabulary has been installed the related policies may be entered. It should be noted that more than one vocabulary can be created; therefore, ensure that the correct one is installed before entering policies or syntax errors will result. The policy is created by entering 'POLICY <policy name>'. This automaticly places the user into the edit mode. A summary of the more common edit commands is listed in Figure 3.6. The policies are entered one per line with a ccmma(,) used as a continuation symbol.

alter <linerange>	modify range of lines
bye/end/quit	leave editor
delete <linerange>	deletes range of lines
find <linerange>	finds line containing the string
help	lists edit commands
insert <lineno|END>	begins inserting lines
load <filename>	loads a modelsource
names <linerange>	inserts variable names
pad <line>	pads with empty lines
parse <ON|OFF>	switches syntax checking on/off
replace <lineno>	replaces line number
scram	leaves editor w/o saving
set <linerange>	establishes a set of lines
stack	temporarily clears edit area
store <filename>	stores a modelsource
type <linerange>	displays a range of lines
unstack	restores the edit area
verify <ON|OFF>	switches verification on/off

Figure 3.6 Summary of REVEAL Edit Commands

J. PROGRAM ENTRY

Once a vocabulary has been installed and the policies established, a program or model must be written to tie them together with the dataset. A program is created by entering 'EDIT'. This puts the user into the edit mode, and a program can be input. Once written this model and the related context must be stored by entering 'STORE <file name>' or it will be lost, including the program, once the current user context is left. To retrieve the context 'LOAD <file name>' is entered. Figure 3.7 shows a simple program for the inventory model, and Figure 3.8 shows the updated matrix once the program is run using the 'EXECUTE' command. The updated matrix is reviewed by entering 'TABLE t1 t10 s1 s7'. This displays terms 1 through 10 and series 1 through 7. The next section will examine in detail the logic used by REVEAL in determining the high limit, low limit, and safety level for item number 2.

K. REVEAL LOGIC ANALYSIS

This section will analyze the sample program line by line, the logic used in policy decision making, and the results of this inventory model for the second item in the sample inventory. The program takes a variable, 'pot.hi.limit', which represents the potential high limit and sets it equal to the demand which is 50. The policies in 'HIGHLIMIT' are now applied. Each policy in 'HIGHLIMIT' is analyzed. If there are logical relations such as 'and' or 'or', the composite statement is analyzed. In this example, endurance equals 43 and the criticality is 2 for part number two. The conditional part of the first policy, 'if endurance is long or criticality is essential', is evaluated on its truth function. As can be seen in Figure 3.9, an endurance of 43 days equates to a degree of truth of

```
 1   pot.hi.limit = demand
 2   Apply ('HIGHLIMIT')
 3   if pot.hi.limit + hi.lim.chng LT 0 then do
 4             high.limit = pot.hi.limit
 5           enddo
 6           else do
 7             high.limit = pot.hi.limit + hi.lim.chng
 8           enddo
 9   pot.lo.limit = .3 * high.limit
10   apply ('LOWLIMIT')
11   if pot.lo.limit + lo.lim.chng LT 0 then do
12             low.limit = pot.lo.limit
13           enddo
14           else do
15             low.limit = pot.lo.limit + lo.lim.chng
16           enddo
17   if criticality EQ 1 then do
18             safe.level = .6 * low.limit
19           enddo
20           else do
21             safe.level = .3 * low.limit
22           enddo
```

Figure 3.7 Sample Inventory Levels Program

	Part1	Part2	Part3	Part4	Part5	Part6	Part7
Demand	5	50	100	3	43	90	14
O&S.T	32	23	17	45	53	72	83
Endurance	30	43	55	20	50	60	90
Criticality	1	2	3	4	5	1	2
High.limit	20	58	100	3	43	105	29
Low.limit	6	8	19	1	20	42	24
Safe.level	4	2	6	0	6	25	7

	Part8	Part9	Part10
Demand	47	84	92
O&S.T	85	90	90
Endurance	80	70	75
Criticality	3	4	5
High.limit	58	92	102
Low.limit	33	42	45
Safe.level	10	13	14

Figure 3.8 Output for Inventory Model

0 for the truth function 'long'. Figure 3.10 shows that a criticality of 2 equates to a degree of truth of .5 for the truth function 'essential'.

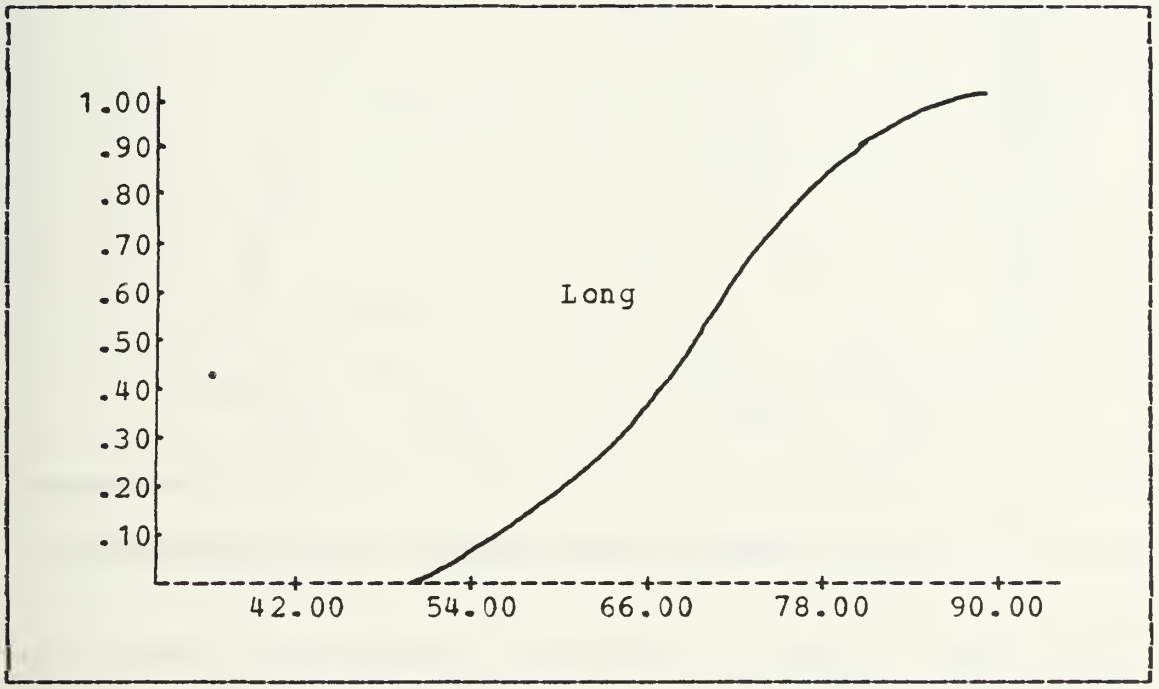


Figure 3.9 Truth Function for Long

Since this is an 'or' situation, the maximum of the two values is taken. Therefore, the degree of truth for the first policy, 'endurance is long or criticality is essential', is .5.

The second policy is analyzed after the first policy. The conditional 'if endurance is normal' is evaluated on its truth function. Figure 3.11 shows that an endurance of 43 days is assigned a degree of truth of .33.

Next, the third policy is analyzed. In this example the conditional 'if endurance is short and criticality is nonessential' is evaluated. Figure 3.12 shows that the degree of truth for an endurance of 43 days is .30 for the

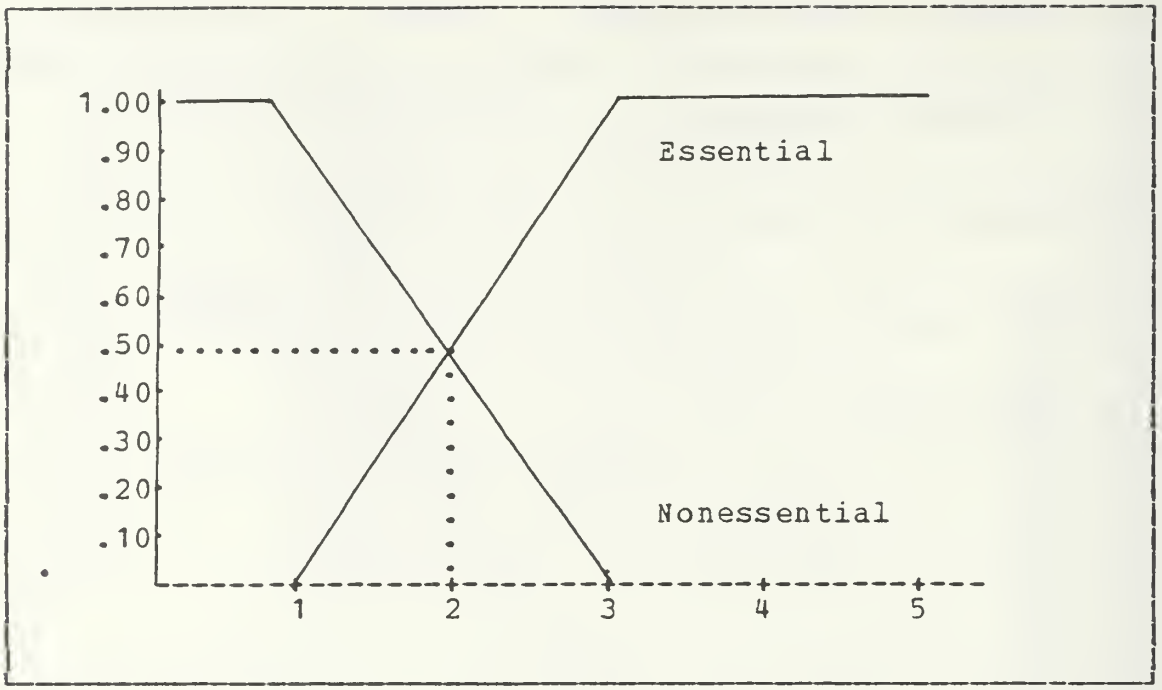


Figure 3.10 Truth Function for Essential and Nonessential

truth function 'short'. Figure 3.10 shows that the degree of truth for a criticality of 2 is .5 for the truth function 'nonessential'. Since this is an 'and' situation, the minimum of the two values is taken as the value of the composite statement. Here, 'endurance is short and criticality is nonessential' evaluates to .30.

The policy with the maximum degree of truth is now selected. Since the first policy's degree of truth is .50, vice .33 for the second policy and .30 for the third policy, it is selected. A value for the consequent proposition 'hi.lim.chng is increased' must now be determined. REVEAL uses the max-min rule of compositional inference. This says that the consequent's degree of truth cannot be any greater then the degree of truth of the antecedent or in this case, .50. Looking at Figure 3.13 it can be seen that a degree of truth of .50 corresponds to an 'increased' value of +8, and

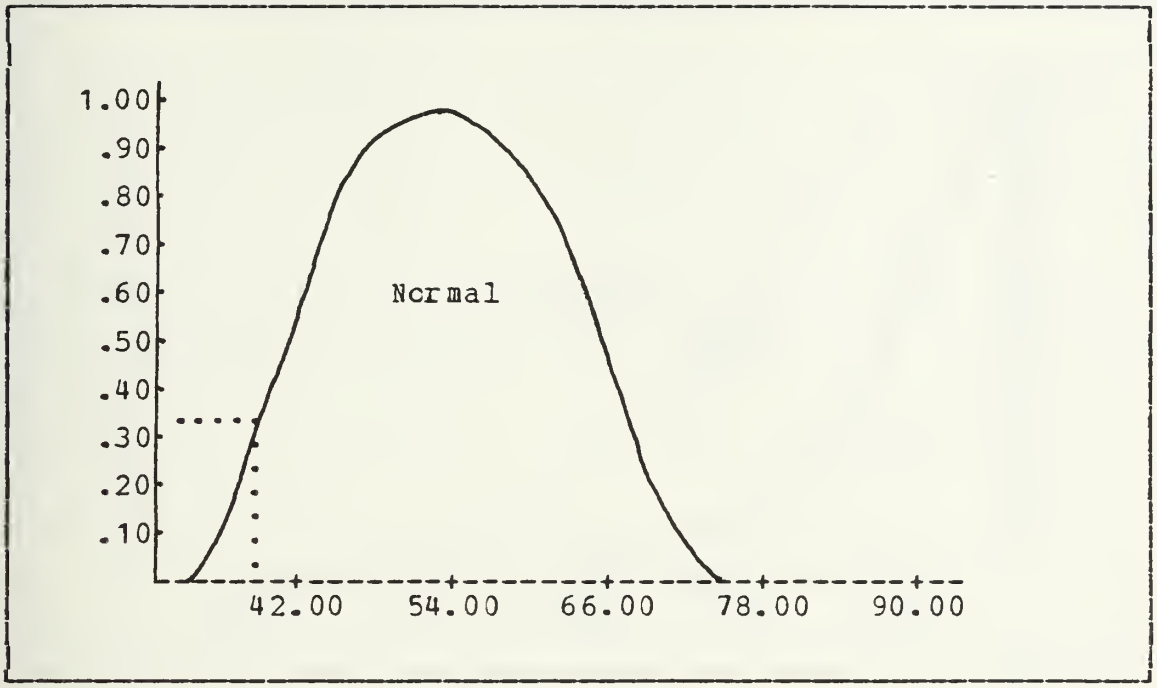


Figure 3.11 Truth Function for Normal

this value is assigned to 'hi.lim.chng'. The policy 'HIGHLIMIT' is now exited. Back in the program, the 'high.limit' is now determined by adding the 'pot.hi.limit' to the 'hi.lim.chng' resulting in a 'high.limit' of 58. Statements 3 and 4 in the program are to prevent a negative high limit from occuring.

The low limit is established based on the high limit. Initially the potential low limit, 'pot.lo.limit', is set to 30% of the 'high.limit'. The policies in 'LOWLIMIT', see Figure 3.3, are now applied. An order and shipping time of 23 days is evaluated in each policy. The first policy determines whether this O&S.T is 'fast', and from Figure 3.14 it can be seen that this equates to a degree of truth of .81. Policy two is evaluated at the same time, and it can be seen that the O&S.T of 23 days equates to a degree of truth of .19 for the truth function 'slow'. Since .81 is the

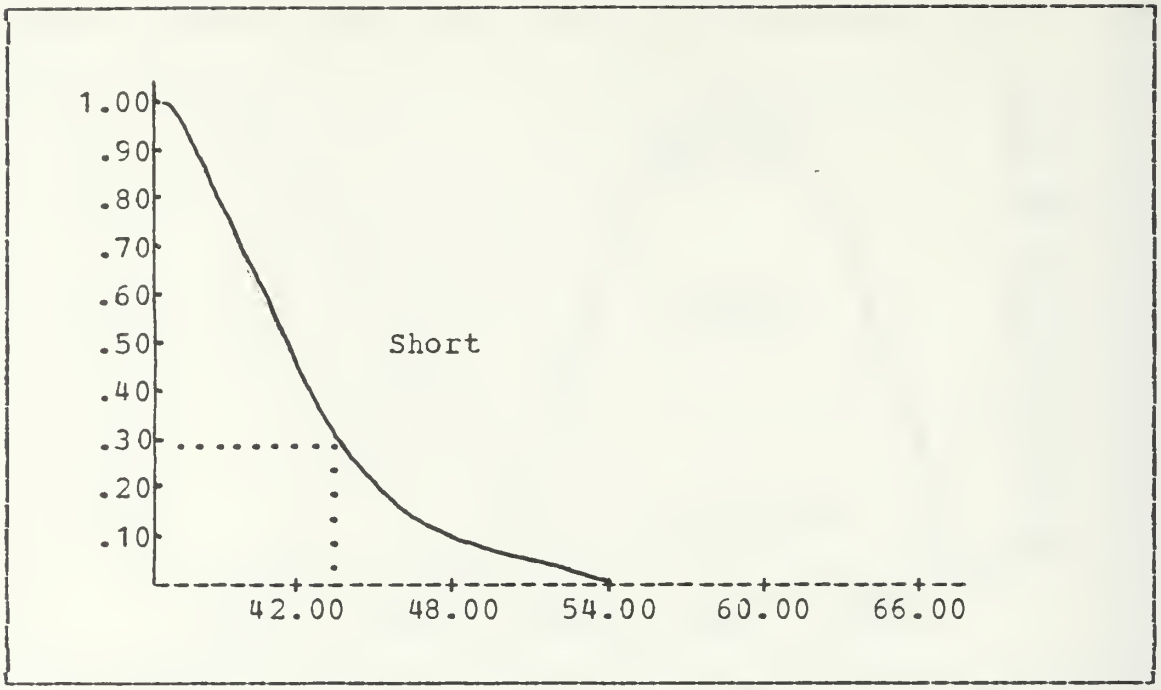


Figure 3.12 Truth Function for Short

'truer' of the two, policy one is the policy that is selected. The consequent of 'somewhat decreased' must now be determined. Since the consequent cannot be 'truer' then the antecedent, here .81, it can be seen from Figure 3.15 that this equates to a value for 'somewhat decreased' of -9. The policy 'LOWLIMIT' is now exited. Back in the program, the 'low.limit' is now established as the sum of 'pot.lo.limit'(17) and 'low.lim.chng'(-9) or 8. Again, statements 11 and 12 are to prevent negative low limits. Finally the safety level ,'safe.level', is established based on the criticality and the low limit. The model run is then complete.

L. SUMMARY

Elimination of artificial boundaries, created by using inventory categories, is the major advantage in using

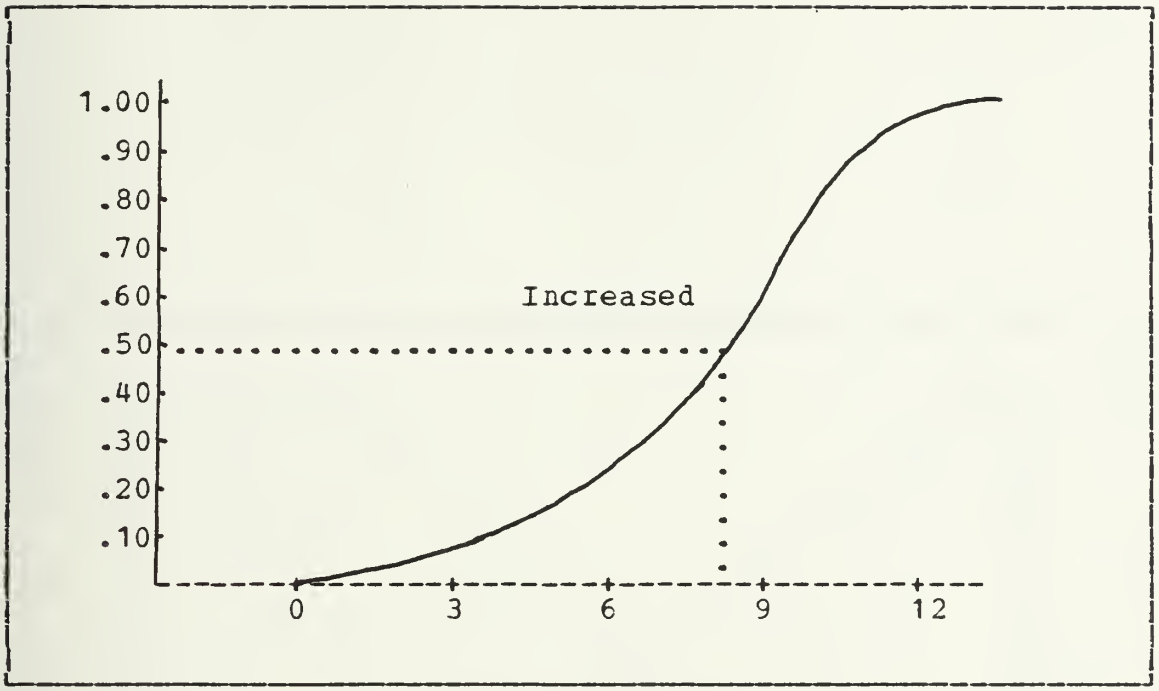


Figure 3.13 Truth Function for Increased

Approximate Reasoning in this inventory model. The procedures used in the Navy determine inventory levels based on distinct categories [Ref. 11]. For instance, Endurance categories of 30, 45, 60, and 75 days are used and Order and Shipping time categories of 0, 30, 75, and 90 days are used. This means that and endurance of 30 days is treated on a par with an endurance of 44 days and an order and shipping time of 75 days is treated the same as an order and shipping time of 89 days. This approach is unrealistic. REVEAL, on the other hand, smoothes out these categories. Through Approximate Reasoning the problem can be modelled in a fashion that is realistic and in line with the way managers reason. The model presents a truer picture of the situation, especially at category extremes. Approximate Reasoning gives the manager more accurate information to aid in the decision making process.

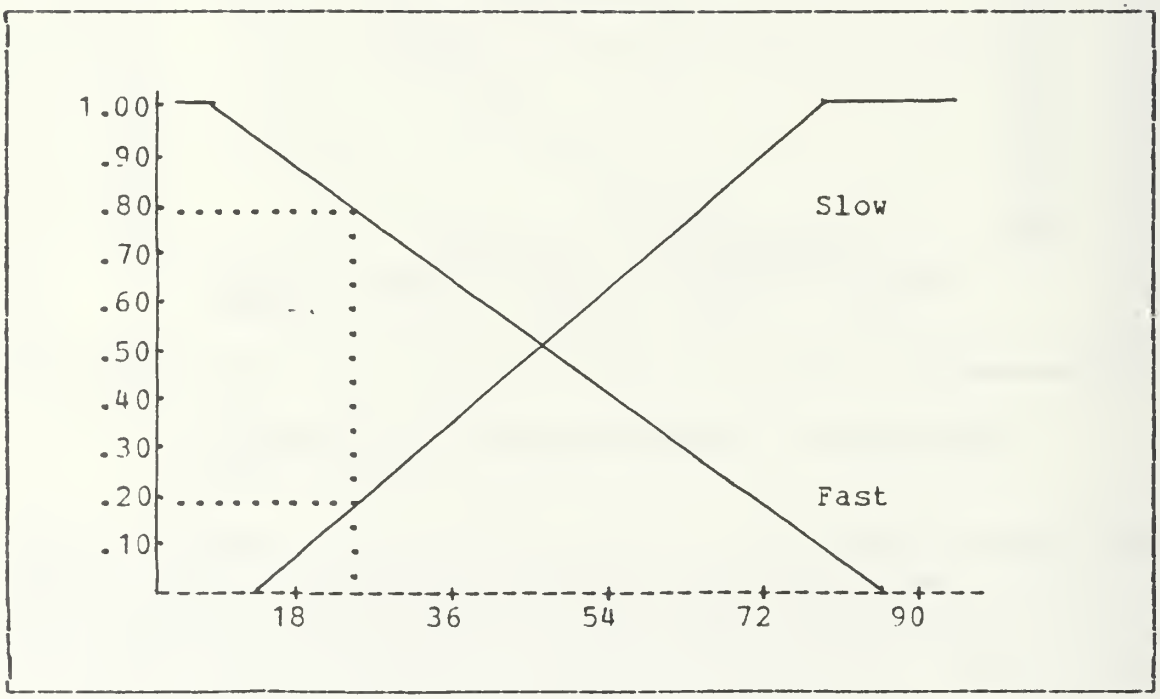


Figure 3.14 Truth Function of Fast and Slow

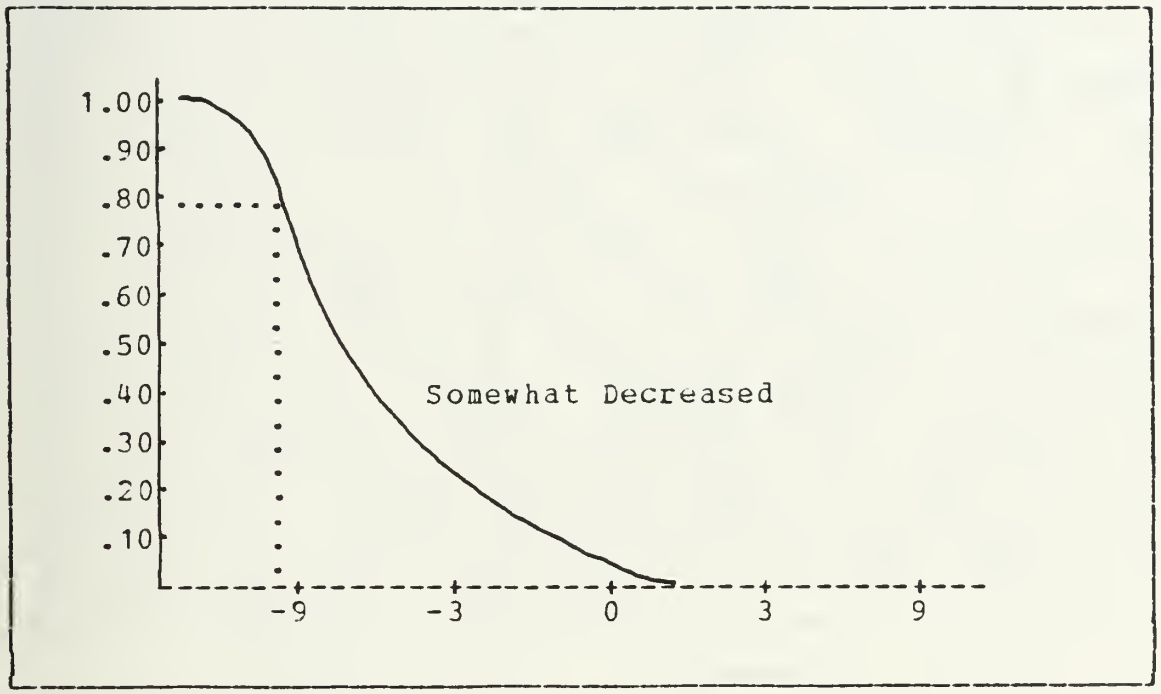


Figure 3.15 Truth Function of Somewhat Decreased

IV. TACTICAL APPLICATIONS IN REVEAL

A. BACKGROUND

Fuzzy Set Theory and REVEAL work well in modelling real world economic and financial systems. Likewise, benefits can be gained by applying Fuzzy Set Theory to certain tactical situations. Military applications exist in the areas of Command, Control and Communications, System and Forces Mix, Effectiveness Measurement, Control and Monitoring of Large Scale Simulations, and Manipulation of Real World Data [Ref. 12]. Dockery [Ref. 13] cites a representative example of the type of problem found within the military. A field commander has two goals when faced with battle: Keep casualties to a minimum and accomplish the mission quickly. In achieving these goals he is faced with several constraints. Stay on the main roads, use minimum armor, keep radio contact to a minimum, and identify bad intelligence are a few. This leaves the field commander with several options. For example, a slow advance with maximum armor, a cross country movement, a quick dash along existing roads, or a coordinated movement with good artillery support.

This example clearly shows that the tactician is faced with many situations involving 'fuzziness'. In this chapter a simplified tactical model will be developed around the following scenario: An aircraft carrier has onboard two types of attack aircraft, A-6's and A-7's, and two types of ordnance, MK82's and Walleyes. The aircraft can carry a variety of weapons in accomplishing their missions. There is a limit in the number of each type of aircraft and in the number of each type of ordnance available for a mission.

There can also exist a great number of missions to which these limited resources can be assigned. The availability of these resources, coupled with the opportunity costs realized by assigning these resources to a particular mission, implies that there is a certain 'cost' or 'value' assigned to each resource. A concern in accomplishing any mission is to minimize these costs. Yet there are certain constraints that must be met. The total numbers of aircraft and ordnance are constraints. Also sufficient resources must be applied to meet the objectives of the mission, ie. destroy the target. To complicate matters there are other factors such as weather, defenses, inbound threat, time of day, target reinforcement, etc. that enter the decision making process. This problem can be approached from a Linear Programming point of view. REVEAL and Approximate Reasoning can be used to determine certain coefficients in the objective function and constraints. By combining the flexibility and 'realness' associated with Fuzzy Sets with the power behind Linear Programming a highly accurate and realistic model can be developed.

B. PROBLEM SETUP

The problem is first set up in the linear programming format. Certain assumptions are made in this example. First, an A-6 can carry only MK82's, whereas an A-7 can carry either MK82's or Walleyes, but not both. If an A-7 carries Walleyes, it can carry only one. These restrictions are imposed solely to simplify this example. The objective function will be to minimize the 'value' or 'cost'

associated with the mission, or stated mathematicly:

Minimize:

('Value' of A-6)X1 +
('Value' of A-7)X2 +
('Value' of A-7)X3 +
{(# of Mk82's/A-6)X1 +
(# of MK82's/A-7)X2}('Value' of MK82) +
{(# of Walleye's/A-7)X3}('Value' of a Walleye)

where:

X1 = Number of A-6's carrying MK82's
X2 = Number of A-7's carrying MK82's
X3 = Number of A-7's carrying Walleyes

The constraints surrounding the availability of aircraft and ordinance and with ensuring sufficient destruction to meet the mission goals are stated as:

1. X1 <= Total number of A-6's available
2. X2 + X3 <= Total number of A-7's available
3. (# of MK82's/A-6)X1 + (# of MK82's/A-7)X2
   <= Total # of MK82's available
4. (# of Walleyes/A-7)X3
   <= Total number of walleyes available
5. (Destruction/A-6 with MK82's)X1 +
   (Destruction/A-7 with MK82's)X2 +
   (Destruction/A-7 with Walleyes)X3
   >= Destruction Required

C. PROBLEM ANALYSIS

Analyzing this problem shows that, in the objective function, the coefficients for number of MK82 per A-6, MK82 per A-7, Walleye per A-7, the value of an A-6, A-7, MK82 and Walleye must all be determined before running the linear programming problem. Likewise, in the constraints it can be seen that, again, the number of MK82 per A-6, MK82 per A-7,

and Walleye per A-7 along with a measure of destruction per A-6 with MK82's, A-7 with MK82's and A-7 with Walleye's must be determined. A total measure of destruction required must also be calculated before proceeding with the linear programming problem. Fuzzy sets and REVEAL will be used to determine values for these coefficients.

1. Objective Function Analysis

The objective function is first evaluated. All coefficients that are to be determined using Approximate Reasoning through REVEAL are examined and qualifiers and policies setup.

a. Aircraft Ordnance Selection

Various factors affect the amount of ordnance carried per aircraft. In this example the type of tactic will be selected from one of four types: Forty degree dive, Laydown, Pop-up, and Loft. Once the tactic is selected, the weapon load will be determined by analyzing the tactic, distance to target, and inbound threat. All three of these parameters are fuzzy sets. The tactic is a fuzzy set that results from the analysis of weather and target defenses (themselves fuzzy sets) and the time of day. Distance to the target is fuzzy in that it can be either close[3] or far. Inbound threat is fuzzy in that it can be either strong or weak.

b. Tactic Selection

Tactic selection is based on the weather, target defense, and time of day. Weather is the first fuzzy parameter and is measured by determining the ceiling, either high or low. Figure 4.1 shows the qualifiers high.ceiling

---

[3]Near can't be used since this is already a REVEAL hedge word.

and low.ceiling. Both qualifiers are defined in the vocabulary as having a low limit of 0 and a high limit of 15,000 feet. The function used for low.ceiling is decline(1000,2500,8000). The function used for high.ceiling is grow(5000,8000,13000).

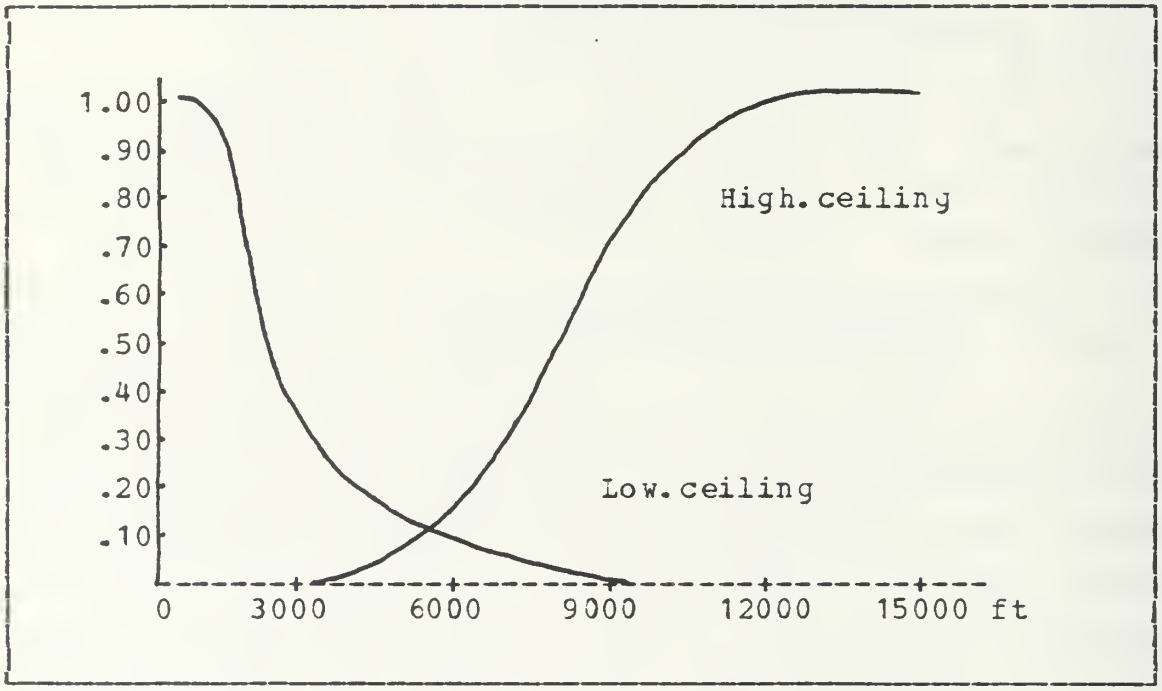


Figure 4.1 Truth Function for High.ceiling and Low.ceiling

The second fuzzy parameter is target defenses. Target defenses can be either low, medium, or high. In this example, the number of anti-aircraft defense batteries will be used to measure target defense. Figure 4.2 shows the qualifiers for low, medium, and high. In all cases the low limit is 0 defense batteries and the high limit is 20 defense batteries. The truth function for low is decline(3,10,25), the truth function for medium is pi(10,8), and the truth function for high is grow(10,14,17).

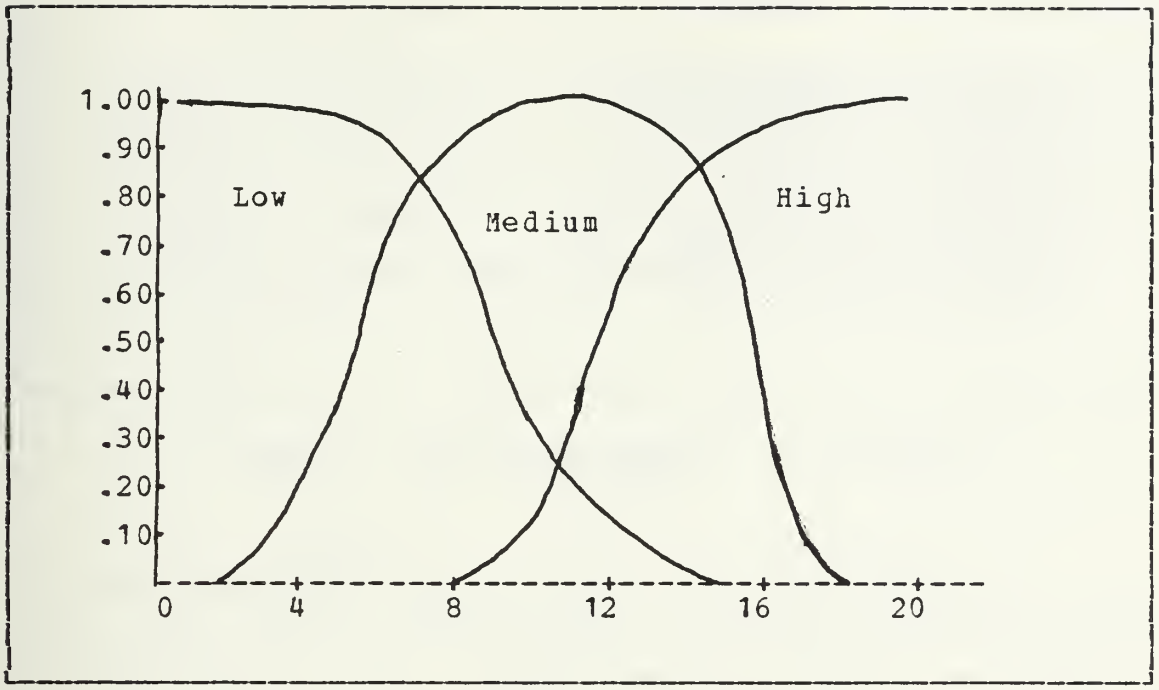


Figure 4.2 High, Medium and Low Defense Batteries

The third parameter in determining the tactic is time of day. This parameter is not fuzzy unless one considers dawn and dusk in which some fuzziness exists. For the purposes of this example day will be defined as 0600 to 1800 and night as 1800 to 0600.

The REVEAL program for this model will ask a series of questions, and the user will input answers via the terminal. Time of day, estimate of enemy defense batteries, and ceiling will be among the questions asked. One of two sets of policies, one for day and one for night, will be used to determine the tactic implemented. The day policies are listed in Figure 4.3. Again it must be remembered that fuzzy set qualifier definitions and the subsequent policy formulations are a completely subjective opinion of the model developer, based on whatever information he has available, and can easily be modified to reflect the feelings of the user.

```
If ceiling is high.ceiling and enemy.batts are low
   then forty.degree is true
If ceiling is high.ceiling and enemy.batts are medium
   then pop.up is true
If ceiling is high.ceiling and enemy.batts are high
   then loft is true
If ceiling is low.ceiling and enemy.batts are high
   then lay.down is true
If ceiling is low.ceiling and enemy.batts are high
   then lay.down is true
```

Figure 4.3 Tactic Selection Policies Daytime.

The result from applying the tactic policies is a degree of truth for each of the four tactics ranging from 0, not true, to 1.0, totally true. The ordnance load is then determined by examining the degree of truth of each tactic along with the distance to target and inbound threat, both of which are input by the user.

Distance is defined as far or close. The low limit is 0 miles and the high limit is 1000 miles. The function for close is decline(50,100,300) and for far is grow(200,500,800). Both are shown in Figure 4.4

The inbound threat is strong or weak and is measured based on the estimated number of enemy aircraft in the target area. The low limit is defined as 0 enemy aircraft and the high limit as 50 enemy aircraft. The function for weak is line(30,10) whereas the function for strong is line(10,40). Both are shown in Figure 4.5

Once the three parameters of tactic, distance, and inbound threat are determined the amount of ordnance for each configuration, ie. MK82/A-6, MK82/A-7, and Walleye/A-7, is determined by applying the appropriate policies. Figure 4.6 shows the ordnance selection policies for MK82's on an

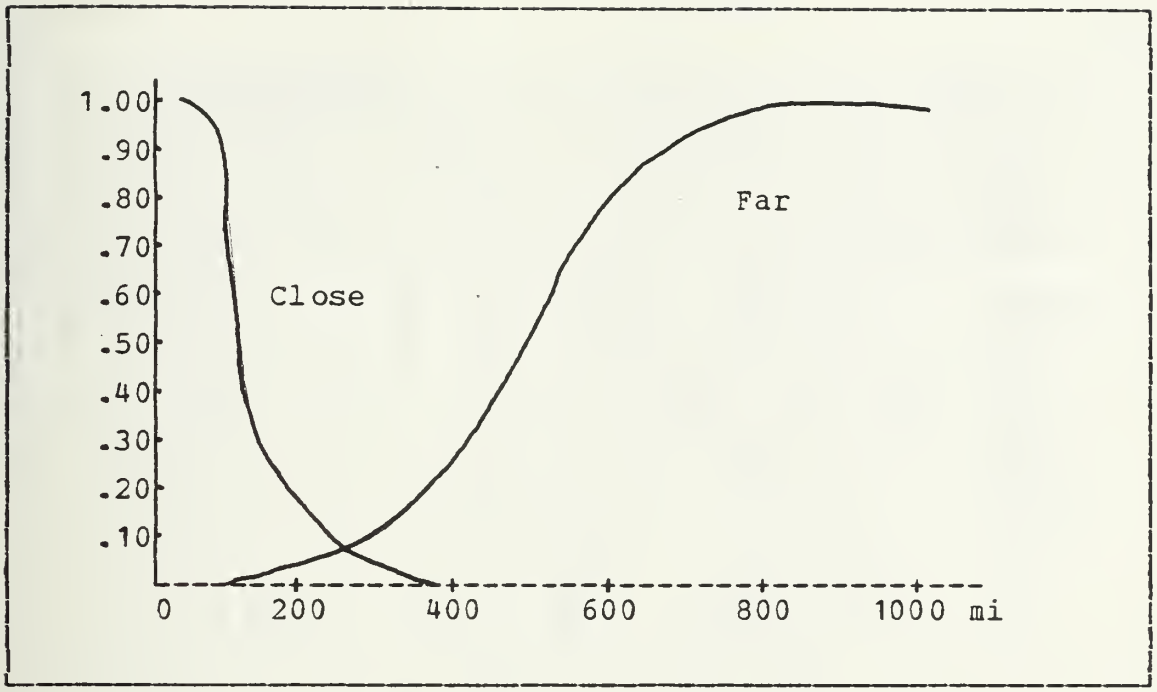


Figure 4.4 Truth Function for Close and Far Distance

A-6. The result from these policy applications is not a hard number but a fuzzy number. For instance, if policy number one was the policy selected because its composite truth value was greatest, the only way exactly 8 MK82's could be selected would be if the composite degree of truth was 1.0. Anything less then 1.0 will produce a weapon load of less then eight MK82's. By carefully formulating policies this selection method results in greater flexibility and versatility in the model. The coefficients of Number of MK82/A-6, Number of MK82's/A-7, and Number of Walleye/A-7 in the objective functicn have now been determined.

c. 'Value' of Aircraft and Ordinance

Determining the value of anything is an extremely difficult and subjective process. In this example, the aircraft involved are A-6's and A-7's. Various

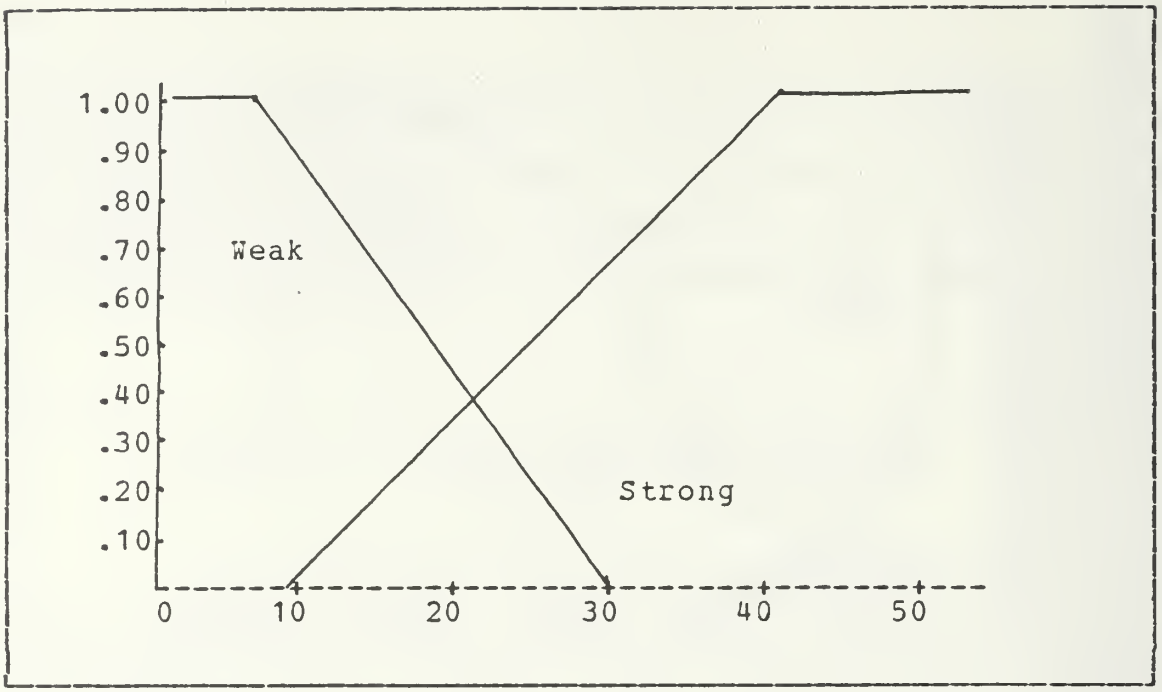


Figure 4.5 Truth Function for Strong and Weak Inbound Threat.

factors such as cost of aircraft, cost to fly per hour, and cost to prepare and support an aircraft for a mission all play roles in determining its 'value'. Similarly, the availability of the aircraft, opportunity costs in assigning an aircraft to one mission vice another, and performance of an aircraft all add or detract from its 'value'. Approximate Reasoning could take these parameters and calculate a value. In keeping this problem simple, however, an A-6 has been assigned a value of 2500 and an A-7 a value of 1500. This value will only be adjusted according to aircraft availability. The availability is determined by comparing the user-inputed, onhand quantity of aircraft to the squadron allowance of aircraft. If availability is low, the 'value' of the aircraft is increased, and if it is high, the 'value' is decreased. Assuming that the sample squadron has 10 A-6's assigned then A6.high will be defined as

```
If forty.degree is true and distance is far and
  inbound.threat is strong then A7.MK82.load is six
If forty.degree is true and distance is far and
  inbound.threat is weak then A7.MK82.load is eight
If forty.degree is true and distance is close and
  inbound.threat is strong then A7.MK82.load is ten
If forty.degree is true and distance is close and
  inbound.threat is weak then A7.MK82.load is twelve
If lay.down is true and distance is far and
  inbound.threat is strong then A7.MK82.load is eight
If lay.down is true and distance is far and
  inbound.threat is weak then A7.MK82.load is ten
If lay.down is true and distance is close and
  inbound.threat is strong then A7.MK82.load is ten
If lay.down is true and distance is close and
  inbound.threat is weak then A7.MK82.load is twelve
If pop.up is true and distance is far and
  inbound.threat is strong then A7.MK82.load is eight
If pop.up is true and distance is far and
  inbound.threat is strong then A7.MK82.load is eight
If pop.up is true and distance is close and
  inbound.threat is strong then A7.MK82.load is ten
If pop.up is true and distance is close and
  inbound.threat is weak then A7.MK82.load is twelve
If loft is true and distance is far and
  inbound.threat is strong then A7.MK82.load is six
If loft is true and distance is far and
  inbound threat is weak then A7.MK82.load is six
If loft is true and distance is close and
  inbound.threat is strong then A7.MK82.load is eight
If loft is true and distance is close and
  inbound.threat is weak then A7.MK82.load is ten
```

Figure 4.6 Ordnance Selection Policies A-7 MK82 Load.

grow(6,8,10) and A6.low will be defined as decline(2,4,6). Likewise, if this airwing has 15 A-7's assigned, A7.high will be defined as grow(5,10,15) and A7.low as decline(5,10,15). Since the 'value' is increased or decreased according to availability, fuzzy sets must be developed for the amount of increase or decrease. A6.value.inc, A6.value.dec, A7.value.inc, and A7.value.dec have been developed for this example.

Availability and value assignments must also be developed for MK82's and Walleyes. This was done in a manner similar to the aircraft 'value' assignments. MK82's

were assigned an initial 'value' of 10 and Walleyes an initial value of 100. These 'values' were increased or decreased according to availability. Again availability was determined by comparing the user input onhand quantity with the squadron allowance, 1500 MK82's and 50 Walleyes, in this example. The 'values' for Value of A6, Value of A7, Value of MK82, and Value of Walleye in the objective function have now been determined.

2. Constraint Analysis

Examining constraint number one shows that only Total A-6's Available needs to be determined. This value was input by the user via the terminal; therefore, it can be assigned. Likewise, the value of Total A-7's Available in constraint number two can be assigned for the same reasons. In constraint number three, values for MK82/A-6 and MK82/A-7 have been determined for the objective function through approximate reasoning, and the same value can be used here. Similarly the value of Walleye/A7 was determined for the objective function, and its value can be assigned here. Constraint number four insures that the destruction deployed on the mission is sufficient enough to destroy the target. Destruction required and destruction capabilities for each type of aircraft configuration, A-6 with MK82's, A-7 with MK82's, and A-7 with Walleyes, must be determined.

a. Destruction Computations

Total destruction required is based on three factors. The first is the type of target. The user selects one of four target types: Petroleum Plant, Building, Bridge, or Airfield. Each has a unit of destruction figure associated with it. These units of destruction, 1000 units, 2000 units, 3000 units and 4000 units respectively, state the amount of destruction required to destroy the target

completely. In this example, the second factor involved is called target properties. This is a measurement on a scale from 0 to 100, input by the user, that determines if the target is soft, moderate, or reinforced. Once target properties are determined, a set of policies is applied and the destruction required is increased or decreased accordingly. The final factor involved is the percent destruction required. This percentage is user input, and a straightforward mathematical calculation of multiplying this percentage times the destruction required, adjusted for target properties, results in a final destruction required figure.

The destruction capabilities of each aircraft configuration is determined by multiplying the ordnance load per aircraft by the destruction capabilities of each type of ordnance. In this example a MK82 is assigned a destruction capability of 100 units and a Walleye is assigned a destruction capability of 1000 units. The ordnance load has already been determined for the objective function. The coefficients for constraint number five have now been determined.

D. LINEAR PROGRAMMING AND PROBLEM SOLUTION

REVEAL and Approximate Reasoning have been used to model a real world situation and determine the coefficient values for a linear programming problem. The linear programming part of this problem can be programmed with REVEAL using the Simplex algorithm. The result is a model tailored to the user's needs and the present situation. The flexibility of the model is self-evident. By combining Operations Research techniques and the theory behind fuzzy sets and REVEAL an effective decision support system can be developed for tactical applications that solves complex, real-world problems.

V. CONCLUSIONS AND RECOMMENDATIONS

A. REVEAL AND APPROXIMATE REASONING

REVEAL and Approximate Reasoning provide an effective way to deal with complex, real-world problems. The concepts behind Fuzzy Set Theory are easily understood and accurately reflect the way humans reason problems through. REVEAL, as the computer implementation of Approximate Reasoning, is a language that can provide managers with realistic and accurate models. The English-like syntax, around which REVEAL operates, is nice to work with once mastered. Approximate Reasoning's similarities to human reasoning allow for great flexibility in data handling. Models can be developed that are easy to operate and understand. The REVEAL system is not, however, without some fault. The availability to obtain printed output is severely restricted. It is also a difficult language to work with and programming expertise is essential. Policies used in modelling quickly become complex. Policy formulation and vocabulary definition, due to their subjectivity, should be conducted by someone that is proficient in the field being modelled. It is imperative that knowledge of the vocabulary used in policy formulation is thorough. As a minimum, familiarity with REVEAL and the application being modelled is essential in creating productive output. The REVEAL system can be an effective decision support system within an organization if the organization understands REVEAL and what to expect from it.

B. SUGGESTED FURTHER RESEARCH

Additional tactical applications and refinement of the one developed in Chapter 4 are areas of further research. Degeneracy and infeasibility within the linear programming segments could be examined. Since the tactical example in this thesis is actually an integer programming problem, an Integer Program to solve the problem should be developed. Currently REVEAL uses a great deal of computer time and other possible hardware configurations should be examined for improved efficiency. Various applications utilizing REVEAL output as input for some other Decision Support System could be explored. A personal computer version of REVEAL is being developed and applications could be identified that would improve field readiness. Finally, non-tactical applications such as personnel management, financial management, inventory and transportation management and other shipboard applications could be identified for further development.

```
TYPE 1 31
  1 : !          Title:   INVENTORY
  2 : !          Author:  D.W. VERHAGEN
  3 : !          Date:    10 FEB 1985
  4 : !          Description:  This program takes data that is manually
  5 : !   entered into a matrix and applies simple policies to obtain
  6 : !   high limits, low limits and safety levels for the sample
  7 : !   inventory using demand, endurance, order and shipping time
  8 : !   and criticality as determining factors.
  9 : !
 10 : !          Context Used:    MODEL1
 11 : !          Vocabulary Used:   MODEL1
 12 : !          Policies Used:   HIGHLIMIT and LOWLIMIT
 13 : !          Command File Used:   NONE
 14 : !          Instructions:  Enter 'load model1' and 'vocabulary model1'
 15 : ! Once context and vocabulary are loaded enter 'execute t1 t10' to
 16 : ! run the sample model.  Entering 'table t1 t10 s1 s7' will display
 17 : ! the adjusted data matrix.
 18 : !
 19 : !
 20 : ! A potential high limit is established
 21 : !
 22 : pot.hi.limit = demand
 23 : !
 24 : ! The policies for establishing the high limit are applied
 25 : !
 26 : apply ('HIGHLIMIT')
 27 : !
 28 : ! This if-then prevents a negative high limit from being established

 29 : !
 30 : if pot.hi.limit + hi.lim.chng LT 0 then do
 31 :          high.limit = pot.hi.limit

*)
```

```
TYPE 31 62
 31 :           high.limit = pot.hi.limit
 32 :         enddo
 33 :         else do
 34 :           high.limit = pot.hi.limit + hi.lim.chng
 35 :         enddo
 36 : !
 37 : ! A potential low limit is established
 38 : !
 39 : pot.lo.limit = .3 * high.limit
 40 : !
 41 : ! The policies for establishing the low limit are applied
 42 : !
 43 : apply ('LOWLIMIT')
 44 : !
 45 : ! This if-then prevents a negative low limit from being established
 46 : !
 47 : if pot.lo.limit + lo.lim.chng LT 0 then do
 48 :           low.limit = pot.lo.limit
 49 :         enddo
 50 :         else do
 51 :           low.limit = pot.lo.limit + lo.lim.chng
 52 :         enddo
 53 : !
 54 : ! This section establishes a safety level
 55 : !
 56 : if criticality EQ 1 then do
 57 :           safe.level = .6 * low.limit
 58 :         enddo
 59 :         else do
 60 :           safe.level = .3 * low.limit
 61 :         enddo

*)
```

```
POLICY HIGHLIMIT
*>TYPE 1 3
  1 : If endurance is long or criticality is essential then the
                hi.lim.chng is increased
  2 : If endurance is normal then hi.lim.chng is nothing
  3 : if endurance is short and criticality is nonessential then the
              hi.lim.chng is decreased

*>QUIT

MODE >POLICY LOWLIMIT
*>TYPE 1 2
  1 : If O&S.T is fast then lo.lim.chng is somewhat decreased
  2 : If O&S.T is slow then lo.lim.chng is somewhat increased

*>QUIT

MODE >
```

DRAW LONG

MODE >

DRAW NORMAL

MODE >

DRAW SHORT

MODE >

DRAW ESSENTIAL

MODE >

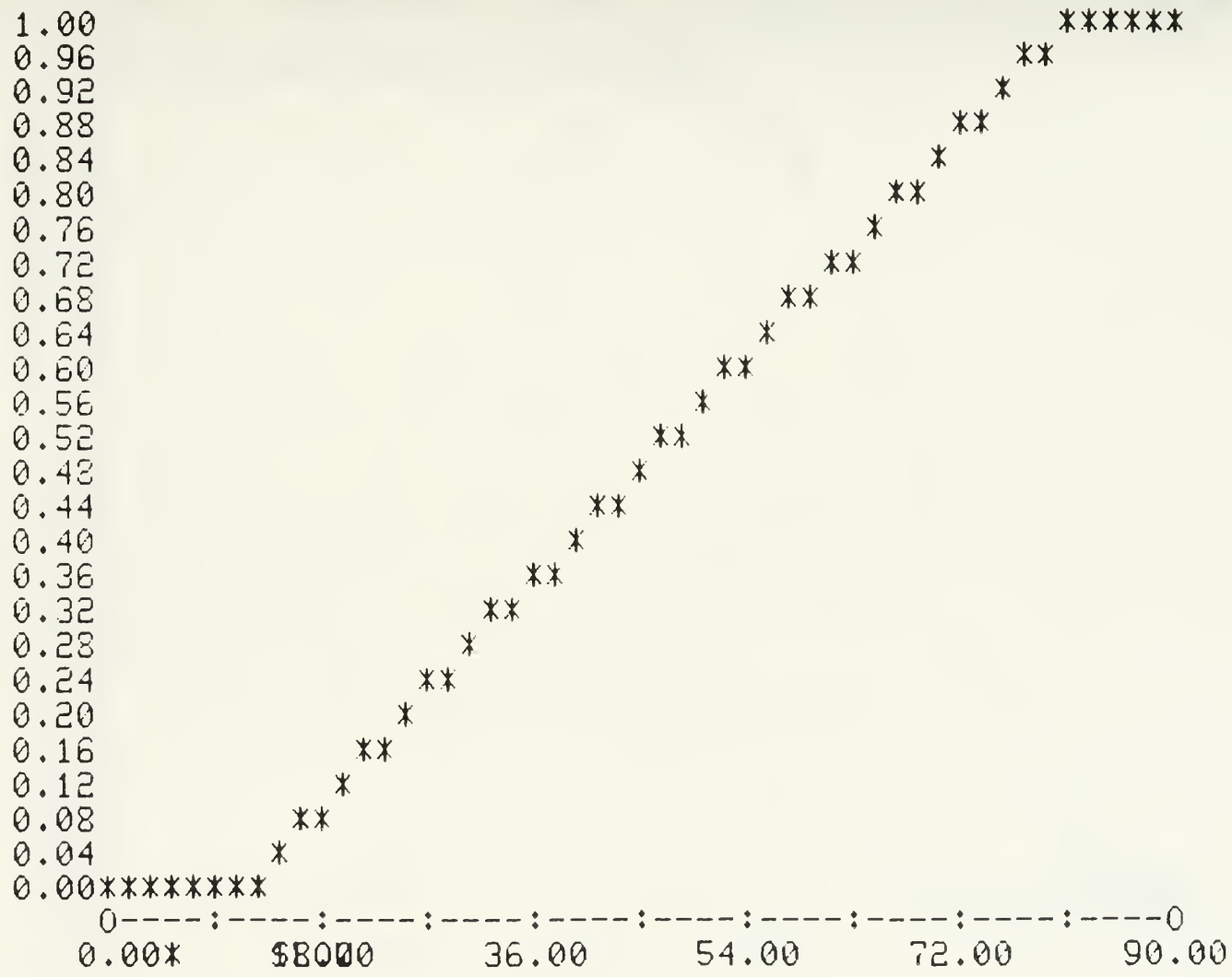
DRAW NONESSENTIAL

MODE >

DRAW FAST

MODE >

DRAW SLOW

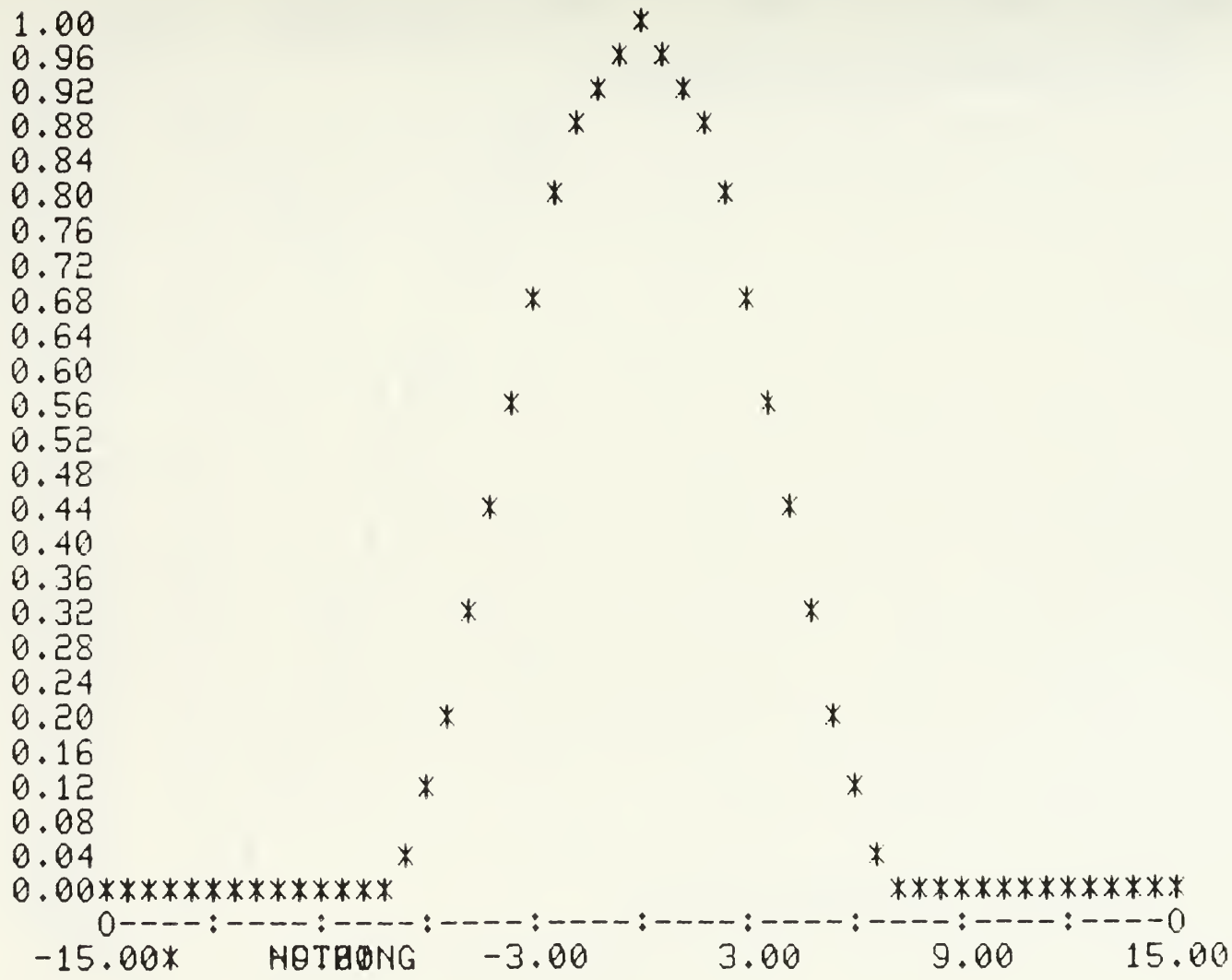


MODE >

DRAW INCREASED

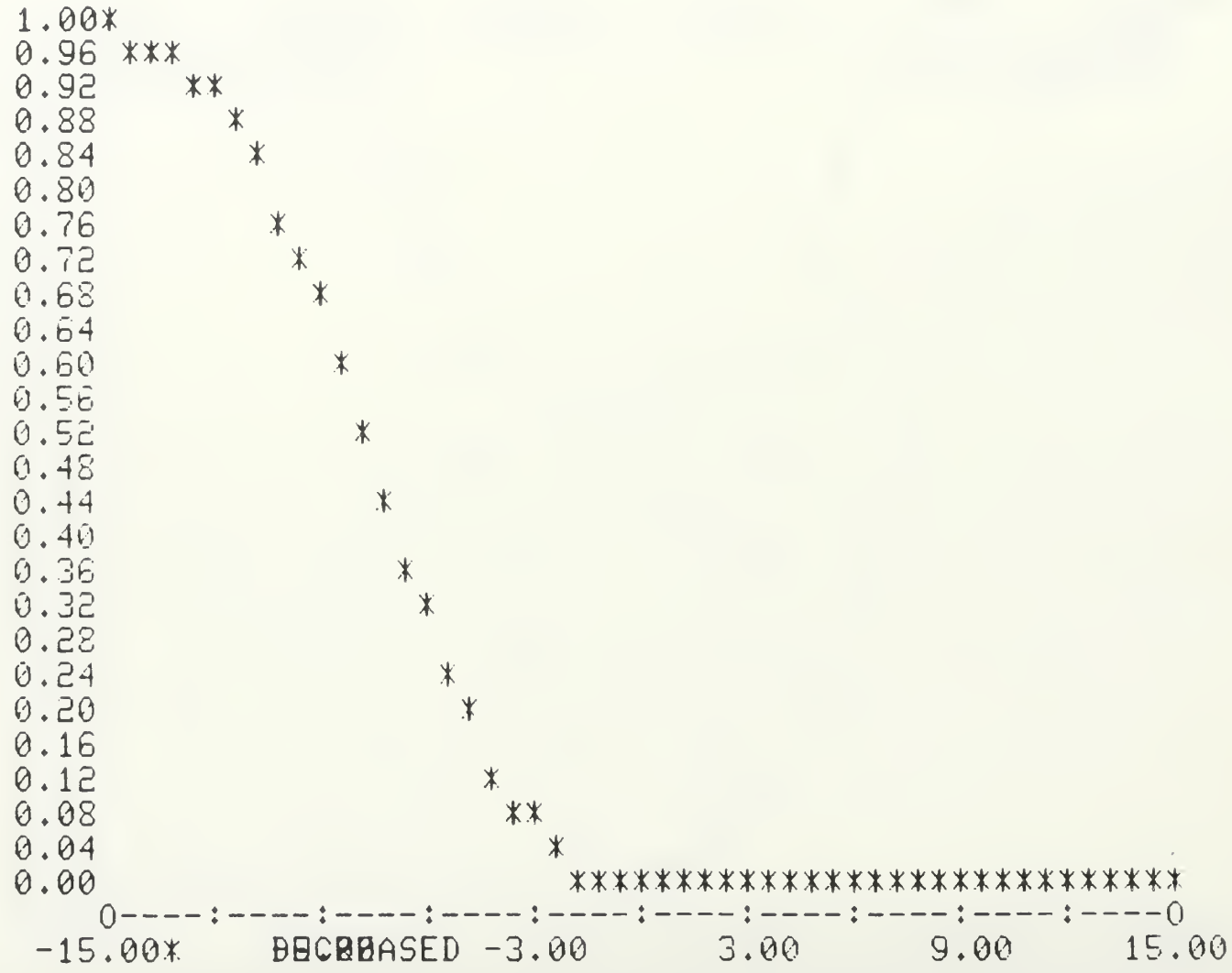
MODE >

DRAW NOTHING



MODE >

DRAW DECREASED



MODE)

draw somewhat decreased

MODE >

TABLE T1 T10 S1 S7

	PART1	PART2	PART3	PART4	PART5	PART6	PART7
DEMAND	5	50	100	3	43	90	14
O&S.T	32	23	17	45	53	72	83
ENDURANCE	30	43	55	20	50	60	90
CRITICALITY	1	2	3	4	5	1	2
HIGH.LIMIT	20	58	100	3	43	105	29
LOW.LIMIT	6	8	19	1	20	42	24
SAFE.LEVEL	4	2	6	0	6	25	7

	PART8	PART9	PART10
DEMAND	47	84	92
O&S.T	85	90	90
ENDURANCE	80	70	75
CRITICALITY	3	4	5
HIGH.LIMIT	58	92	102
LOW.LIMIT	33	42	45
SAFE.LEVEL	10	13	14

MODE >

```
type 1 100
  1 :
  2 : !       Name: Tactic
  3 : !       Author: D.W. Verhagen
  4 : !       Date: 21 FEB 1985
  5 : !       Date Last Updated: 13 MAR 1985
  6 : ! Summary: This is the command file for the tactic
  7 : ! program. It calls the tactic context and loads
  8 : ! the vocabulary for the sample run. Execution is
  9 : ! on t10 to allow the 'sermin' function to be used
 10 : ! in the Linear Programming segment.
 11 : !
 12 : !
 13 : type 1 100
 14 : load doug:tactic
 15 : vocabulary doug:tactic
 16 : decims 0
 17 : execute t10 t10

*>
```

```
type 1 30
  1 : ! Title: Tactic
  2 : ! Author: D. W. Verhagen
  3 : ! Date: 19 FEB 1985
  4 : ! Date Last Updated: 6 MAR 1985
  5 : ! Description: This program takes various user input and
  6 : !     computes the coefficients for a linear programming
  7 : !     problem.  The problem is to minimize cost subject to
  8 : !     constraints on aircraft availability, weapon avail-
  9 : !     ability, and destruction capability.  The user simply
 10 : !     answers all questions as they appear.
 11 : ! Command file used: Doug:tactic
 12 : ! Context used: Doug:tactic
 13 : ! Vocabulary used: Doug:tactic
 14 : ! Policies used:  Destruction
 15 : !                 A6.avail
 16 : !                 A7.avail
 17 : !                 MK82.avail
 18 : !                 Wall.avail
 19 : !                 Nite.policy
 20 : !                 Day.policy
 21 : !                 A6.MK82.load
 22 : !                 A7.MK82.load
 23 : !                 A7.Wall.load
 24 : ! Directions:  Enter "Doug:tactic" after logging on and
 25 : !              entering REVEAL
 26 : !
 27 : !
 28 : target:
 29 :           print ('Please enter target type: ')
 30 :           print (' ')

*)
```

```
type 31 61
 31 :            print (' Petroleum Plant --------- 0  ')
 32 :            print (' Building ----------------1  ')
 33 :            print (' Bridge ------------------ 2  ')
 34 :            print (' Air Field --------------- 3  ')
 35 :            target = tty
 36 :            if target GT 3 then goto target:
 37 :            if target EQ 0 then destruction = 1000
 38 :            if target EQ 1 then destruction = 2000
 39 :            if target EQ 2 then destruction = 3000
 40 :            if target EQ 3 then destruction = 4000
 41 : props:
 42 :            print ('Please enter target properties on a scale ')
 43 :            print ('from 0 (soft) to 100 (reinforced):  ')
 44 :            target.prop = tty
 45 :            if target.prop GT 100 then goto props:
 46 :            apply ('DESTRUCTION')
 47 : percent:
 48 :            print ('Please enter percentage damage required: ')
 49 :            print ('   (format is XX   %)       ')
 50 :            percentage = tty
 51 :            if percentage   GT 100 then goto percent:
 52 : !
 53 :            destruc.rqd = (percentage/100) * (destruction + destruc.ch
ng)
 54 : !
 55 : !
 56 : !
 57 : a6.onhd:
 58 :            print ('What is the total number of A6s available? ')
 59 :            print ('     (Number must be less then 10)          ')
 60 :            a6.available = tty
 61 :            if A6.available GT 10 then goto A6.onhd:

*)
```

```
type 62 90
 62 : a7.onhd:
 63 :            print ('What is the total number of A7s available?  ')
 64 :            print ('   (Number must be less then 20)            ')
 65 :            a7.available = tty
 66 :          if A7.available GT 20 then goto A7.onhd:
 67 : 82.onhd:
 68 :            print ('What is the total number of MK82s available?  ')
 69 :            print ('      (Number must be less then 1500)          ')
 70 :            mk82.avail = tty
 71 :            if mk82.avail GT 1500 then goto 82.onhd:
 72 : wl.onhd:
 73 :            print ('What is the total number of Walleyes available? ')

 74 :            print ('    (Number must be less then 40)               ')
 75 :            walleye.ava = tty
 76 :            if walleye.ava GT 40 then goto wl.onhd:
 77 : !
 78 : ! Values are adjusted based upon availability and allowance
 79 : !
 80 :            apply ('A6.AVAIL')
 81 :                 tot.a6.val = a6.value + a6.val.chng
 82 :            apply ('A7.AVAIL')
 83 :                 tot.a7.val = a7.value + a7.val.chng
 84 :            apply ('MK82.AVAIL')
 85 :                 tot.mk82.val = mk82.value + m82.val.chng
 86 :            apply ('WALL.AVAIL')
 87 :                 tot.wall.val = walleye.valu + wal.val.chng
 88 : !
 89 : timeday:
 90 :          Print ('Please enter the scheduled time of the bomb run: ')


*)
```

```
type 90 110
 90 :           Print ('Please enter the scheduled time of the bomb run: ')

 91 :            print ('      (Format is military time, XXXX)            ')

 92 :            time.of.day = tty
 93 :            if time.of.day GT 2400 then goto timeday:
 94 : weather:
 95 :            print ('What is todays ceiling at time of bomb run?      '
)
 96 :            print ('     (format is XXXXX ft.)                        '
)
 97 :            ceiling = tty
 98 :            if ceiling GT 50000 then goto weather:
 99 : batrys:
100 :            print ('What is estimated number of enemy defense batterie
s?')
101 :            print ('          (Number must be less then 20'
  ')
102 :            enemy.batts = tty
103 :            if enemy.batts GT 20 then goto batrys:
104 : !
105 : ! Tactic is determined, policies selected are based upon
106 : !     the time of day.
107 : !
108 :            if time.of.day LT 600 then apply('NITE.POLICY')
109 :            if time.of.day GT 1800 then apply('NITE.POLICY')
110 :            if time.of.day GE 600 and time.of.day LE 1800 then
                          apply('DAY.POLICY')

*>
```

```
type 110 140
110 :             if time.of.day GE 600 and time.of.day LE 1800 then
                                  apply('DAY.POLICY')
111 : !
112 : !
113 : dist:
114 :             print ('What is distance to target in miles?  ')
115 :             print ('    (Number must be less then 1000)   ')
116 :             distance = tty
117 :             if distance GT 1000 then goto dist:
118 : threat:
119 :             print ('What is estimated inbound threat?      ')
120 :             print ('  (In number of enemy aircraft)        ')
121 :             print ('  (Number must be less then 40)        ')
122 :             inbound.thrt = tty
123 :             if inbound.thrt GT 40 then goto threat:
124 : !
125 : ! The weapon load is determined.
126 : !
127 :             apply('A6.MK82.LOAD')
128 :             apply('A7.MK82.LOAD')
129 :             apply('A7.WALL.LOAD')
130 : !
131 : !
132 :             a6.mk82.dstr = 10 * a6.mk82.load
133 :             a7.mk82.dstr = 10 * a7.mk82.load
134 :             a7.wall.dstr = 100 * a7.wall.load
135 : !
136 : ! Values consolidated to facilitate equation manipulation.
137 : !
138 : a = tot.a6.val + (tot.mk82.val * a6.mk82.load)
139 : b = tot.a7.val + (tot.mk82.val * a7.mk82.load)
140 : c = tot.a7.val + (tot.wall.val * a7.wall.load)

*)
```

```
type 140 170
140 : c = tot.a7.val + (tot.wall.val * a7.wall.load)
141 : !
142 : ! Display the suggested tactic after finding it
143 : !
144 : M = 1
145 : repeat until M LE 0
146 : if loft GE M and flag2 EQ 0 then do
147 :      print ('Suggested Tactic is Loft')
148 :      flag2 = 1
149 :    enddo
150 : if lay.down GE M and flag2 EQ 0 then do
151 :     print ('Suggested Tactic is Laydown')
152 :     flag2 = 1
153 :   enddo
154 : if pop.up GE M and flag2 EQ 0 then do
155 :     print ('Suggested Tactic is Pop-up')
156 :     flag2 = 1
157 :   enddo
158 : if forty.degree GE M and flag2 EQ 0 then do
159 :     print ('Suggested Tactic is Forty Degree Dive')
160 :     flag2 = 1
161 :   enddo
162 : M = M - .01
163 : endrep
164 : !
165 : ! print the weapon load per aircraft
166 : !
167 : print (' ')
168 : print (' ')
169 : print ('Suggested A6 MK82 load is ', a6.mk82.load)
170 : print (' ')

*)
```

```
type 170 200
170 : print (' ')
171 : print ('Suggested A7 MK82 load is ', a7.mk82.load)
172 : print (' ')
173 : print ('Suggested A7 Walleye load is ', a7.wall.load)
174 : print (' ')
175 : !
176 : print ('Hit enter to continue')
177 : flag = tty
178 : !
179 : !
180 : ! Fuzzy calculations are complete, LP part commences.
181 : !
182 : !
183 : print ('Linear Programming Data is: ')
184 : print (' ')
185 : print (' ')
186 : print ('Objective Function: ')
187 : print (' ')
188 : print ('Minimize:')
189 : print (' ')
190 : print (a, ' X11 + ',b, ' X21 +', c, ' X22')
191 : print (' ')
192 : print ('Subject to:')
193 : print (' ')
194 : print('X11  <  ',a6.available)
195 : print (' ')
196 : print('X21 + X22  <  ',a7.available)
197 : print (' ')
198 : print (a6.mk82.load, ' X11 +',a7.mk82.load, ' X21 <',mk82.avail)
199 : print (' ')
200 : print (a7.wall.load, ' X22 <', walleye.ava)

*>
```

```
type 200 225
200 : print (a7.wall.load, ' X22 (', walleye.ava)
201 : print (' ')
202 : print (a6.mk82.dstr, 'X11 +', a7.mk82.dstr, 'X21 +', a7.wall.dstr,

            'X22 )' , destruc.rqd)
203 : !
204 : print ('hit enter to continue')
205 : flag = tty
206 : !
207 : !
208 : ! LOAD THE LP MATRIX HERE
209 : !   The matrix is loaded with the objective function negated
210 : !       and the problem treated as a maximization.
211 : !
212 : !
213 : series(ob.func,1) = 1
214 : SERIES(OB.FUNC,2) = A
215 : SERIES(OB.FUNC,3) = B
216 : series(ob.func,4) = C
217 : series(ob.func,5) = 0
218 : series(ob.func,6) = 0
219 : series(ob.func,7) = 0
220 : series(ob.func,8) = 0
221 : series(ob.func,9) = 0
222 : series(ob.func,10) = MEGA
223 : series(ob.func,11) = 0
224 : !
225 : !

*)
```

```
type 225 255
225 : !
226 : series(a6.tot.con,1) = 0
227 : series(a6.tot.con,2) =1
228 : series(a6.tot.con,3) = 0
229 : series(a6.tot.con,4) = 0
230 : series(a6.tot.con,5) = 1
231 : series(a6.tot.con,6) = 0
232 : series(a6.tot.con,7) = 0
233 : series(a6.tot.con,8) = 0
234 : series(A6.tot.con,9) = 0
235 : series(a6.tot.con,10) = 0
236 : series(a6.tot.con,11) = A6.available
237 : !
238 : series(a7.tot.con,1) = 0
239 : series(a7.tot.con,2) =0
240 : series(a7.tot.con,3) = 1
241 : series(a7.tot.con,4) = 1
242 : series(a7.tot.con,5) = 0
243 : series(a7.tot.con,6) = 1
244 : series(a7.tot.con,7) = 0
245 : series(a7.tot.con,8) = 0
246 : series(a7.tot.con,9) = 0
247 : series(a7.tot.con,10) = 0
248 : series(a7.tot.con,11) = a7.available
249 : !
250 : series(mk82.tot.con,1) = 0
251 : series(mk82.tot.con,2) = A6.MK82.load
252 : series(mk82.tot.con,3) = A7.MK82.load
253 : series(mk82.tot.con,4) = 0
254 : series(mk82.tot.con,5) = 0
255 : series(mk82.tot.con,6) = 0

*)
```

```
type 255 285
255 : series(mk82.tot.con,6) = 0
256 : series(mk82.tot.con,7) = 1
257 : series(mk82.tot.con,8) = 0
258 : series(mk82.tot.con,9) = 0
259 : series(mk82.tot.con,10) = 0
260 : series(mk82.tot.con,11) = MK82.avail
261 : !
262 : series(wall.tot.con,1) = 0
263 : series(wall.tot.con,2) = 0
264 : series(wall.tot.con,3) = 0
265 : series(wall.tot.con,4) = A7.WALL.load
266 : series(wall.tot.con,5) = 0
267 : series(wall.tot.con,6) = 0
268 : series(wall.tot.con,7) = 0
269 : series(wall.tot.con,8) = 1
270 : series(wall.tot.con,9) = 0
271 : series(wall.tot.con,10) = 0
272 : series(wall.tot.con,11) = Walleye.ava
273 : !
274 : series(dstr.con,1) = 0
275 : series(dstr.con,2) = A6.MK82.dstr
276 : series(dstr.con,3) = A7.MK82.dstr
277 : series(dstr.con,4) = A7.WALL.dstr
278 : series(dstr.con,5) = 0
279 : series(dstr.con,6) = 0
280 : series(dstr.con,7) = 0
281 : series(dstr.con,8) = 0
282 : series(dstr.con,9) =-1
283 : series(dstr.con,10) = 1
284 : series(dstr.con,11) = destruc.rqd
285 : !

*)
```

```
Type 285 310
285 : !
286 : !
287 : series(bas,2) = 5
288 : series(bas,3) = 6
289 : series(bas,4) = 7
290 : series(bas,5) = 8
291 : series(bas,6) = 10
292 : !
293 : !
294 : ! Use Simplx Method to Determine Correct Weapon mix
295 : !
296 : itcount = 0   !establish an iteration counter
297 : !
298 : ! clear MEGA (a1) out of matrix
299 : !
300 : M = 1
301 : repeat until M GT numvar + 1
302 :       series(ob.func,M) = series(ob.func,M) -
                    (series(dstr.con,M) * MEGA)
303 : endrep; M = M + 1
304 : !
305 : ! Iterate until all objective function coefficients are driven posit
ive
306 : !
307 : repeat until sermin(objective,numvar) GE 0 or itcount EQ 25
308 :       itcount = itcount + 1
309 : !
310 : ! Locate the pivot column if it exists

*>
```

```
type 310 340
310 : ! Locate the pivot column if it exists
311 : !
312 :       I = 1
313 :           repeat until term(objective,I) EQ sermin(objective,numva
r)
314 :       endrep; I = I + 1
315 : !
316 : ! Locate the pivot element within the column
317 : !
318 : J = 2; XMAX = 10 * MEGA
319 : repeat until J GT numcon
320 :       X = series(J,RHS)/series(J,I)
321 :           if X GT 0 and X LT XMAX then XMAX = X; K = J
322 :       endrep; J = J + 1
323 : !
324 : ! Update the basis row
325 : !
326 :       term(basis,K) = I
327 : !
328 : ! Perform the pivot operation
329 : !
330 :       M = 1
331 :       repeat until M GT (numvar + 1)
332 :          if M NE I then series(K,M) = series(K,M)/series(K,I)
333 :       endrep; M = M + 1
334 : !
335 :       series(K,I) = 1
336 : !
337 :       M = 1
338 :       repeat until M GT numcon
339 :          N = 1
340 :          repeat until N GT (numvar + 1)

*)
```

```
type 340 370
340 :            repeat until N GT (numvar + 1)
341 :                if N NE I and M NE K then series(M,N) =
                                 series(M,N) - series(K,N) * series (M,I)
342 :            endrep; N = N + 1
343 :            if M NE K then series(M,I) = 0
344 :        endrep; M = M + 1
345 : !
346 : !
347 : if itcount GE 20 then flag3 = 1
348 : !
349 : endrep
350 : !
351 : !
352 : ! Pivot is completed, major loop completed
353 : !
354 : if flag3 EQ 1 then do
355 :     print ('No Feasible Solution Obtained')
356 :     print ('*********************************')
357 : !
358 :
359 : enddo
360 : else do
361 : !
362 : !Format and print the results
363 : !
364 : print ('Solution obtained after',itcount like 'XXX',' iterations')
365 : print (' ')
366 : print('The value of the OBJECTIVE FUNCTION is', series(1,rhs))
367 : print (' ')
368 : print ('The BASIC VARIABLES and their values are: ')
369 : print (' ')
370 : I = 2

*)
```

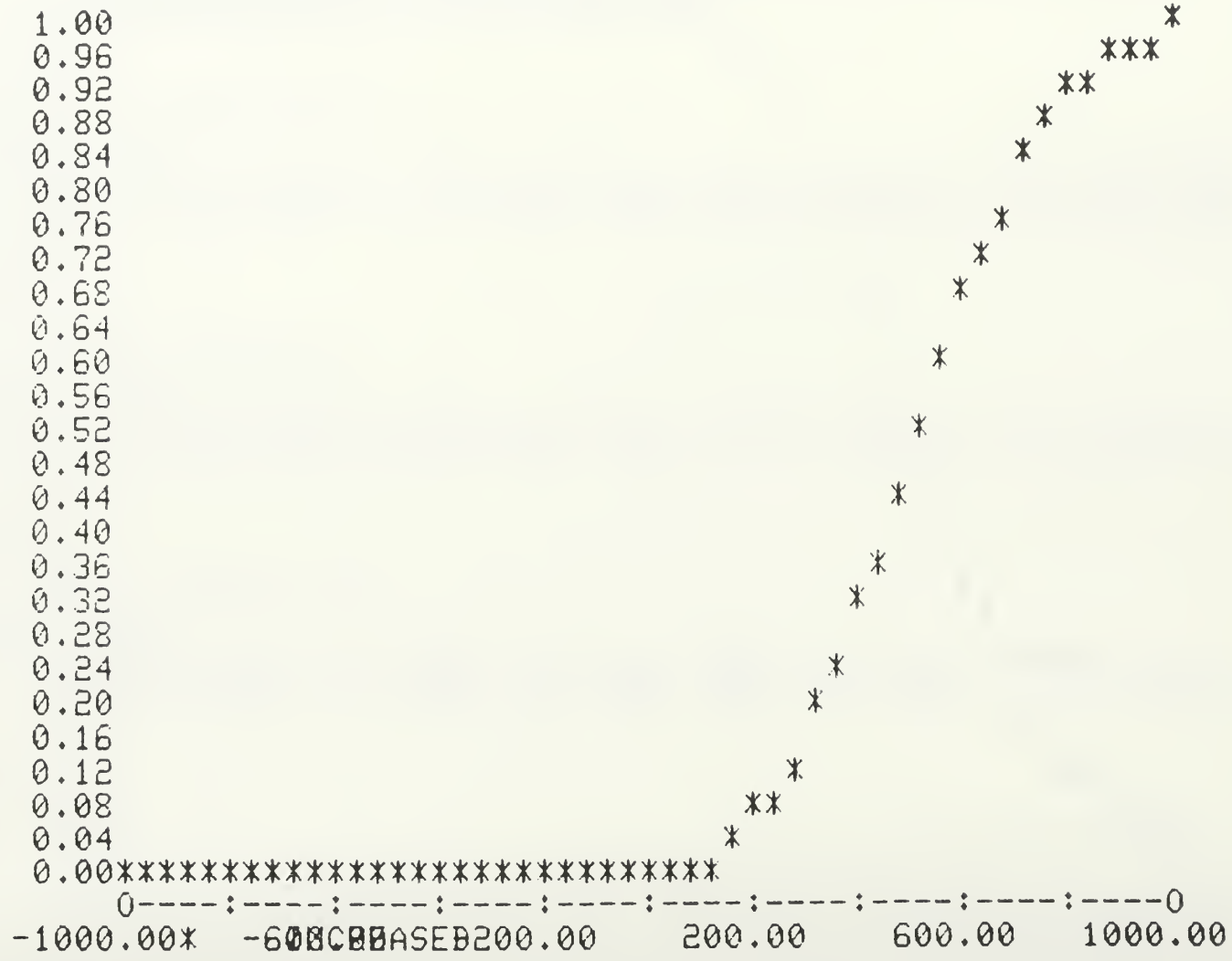
```
type 370 400
370 : I = 2
371 : repeat until I GT numcon
372 : print('    X(',term(basis,I) - 1 like 'XX',')      ',series(I,RHS))
373 : endrep; I = I + 1
374 : print (' ')
375 : enddo
376 : !
377 : !
378 : exit

*)
```

```
policy destruction
*>type 1 100
  1 : if target.prop is reinforced then destruc.chng is increased
  2 : if target.prop is moderate then destruc.chng is nothing
  3 : if target.prop is soft then destruc.chng is decreased

*>quit

MODE >policy a6.avail
*>type 1 100
  1 : if a6.available is a6.low then a6.val.chng is a6.increased
  2 : if a6.available is a6.high then a6.val.chng is a6.decreased

*>quit

MODE >policy a7.avail
*>type 1 100
  1 : if a7.available is a7.low then a7.val.chng is a7.increased
  2 : if a7.available is a7.high then a7.val.chng is a7.decreased

*>quit

MODE >policy mk82.avail
*>type 1 100
  1 : if mk82.avail is mk82.low then m82.val.chng is mk82.incrsd
  2 : if mk82.avail is mk82.high then m82.val.chng is mk82.decrsd

*>quit

MODE >
```

draw reinforced

MODE >

draw increased

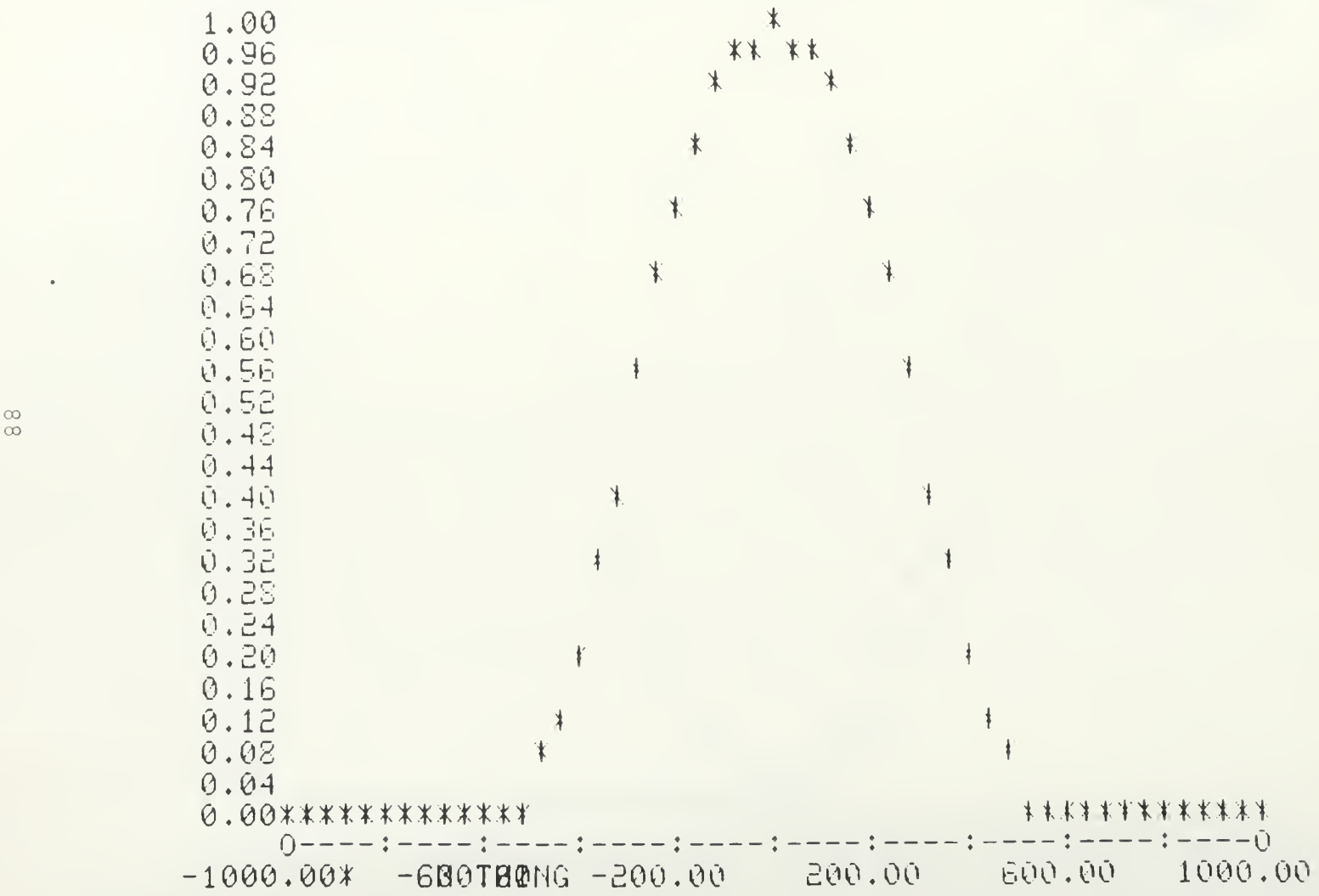


MODE >

draw moderate

MODE :

draw nothing



MODE >

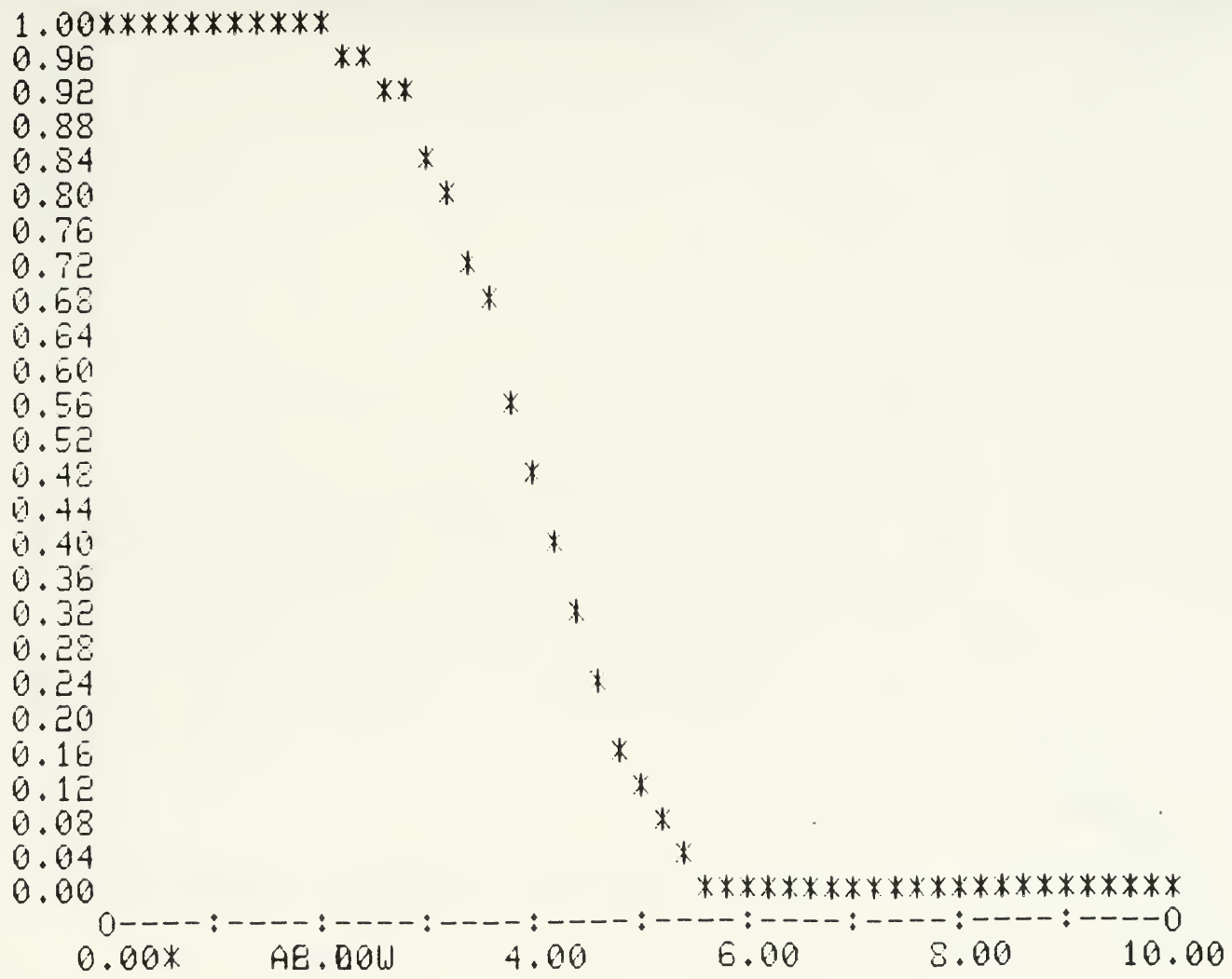
draw soft

MODE >

draw decreased

MODE >

raw a6.low



MODE >

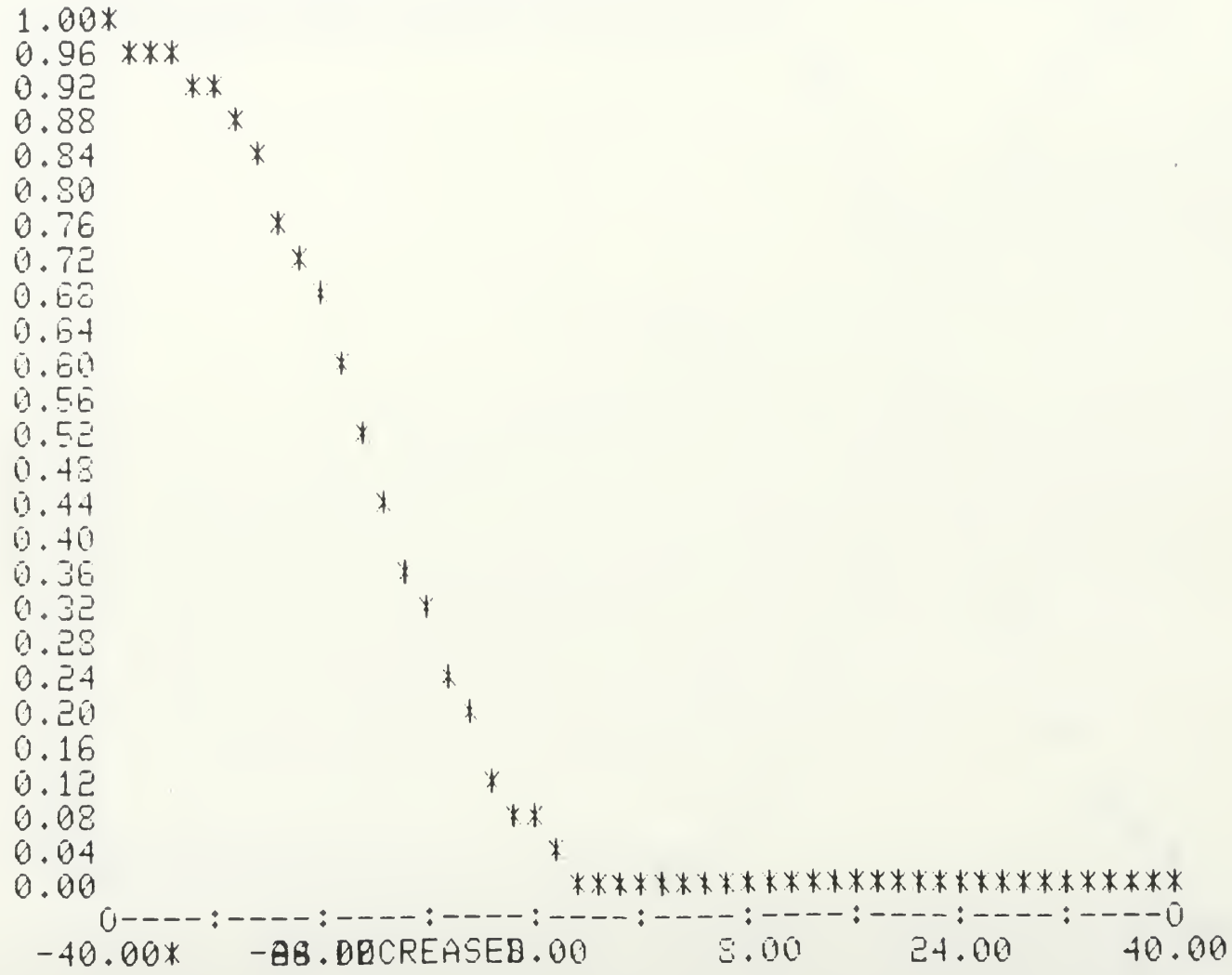
```
draw a6.increased
```


```
MODE >
```

```
draw a6.high
```

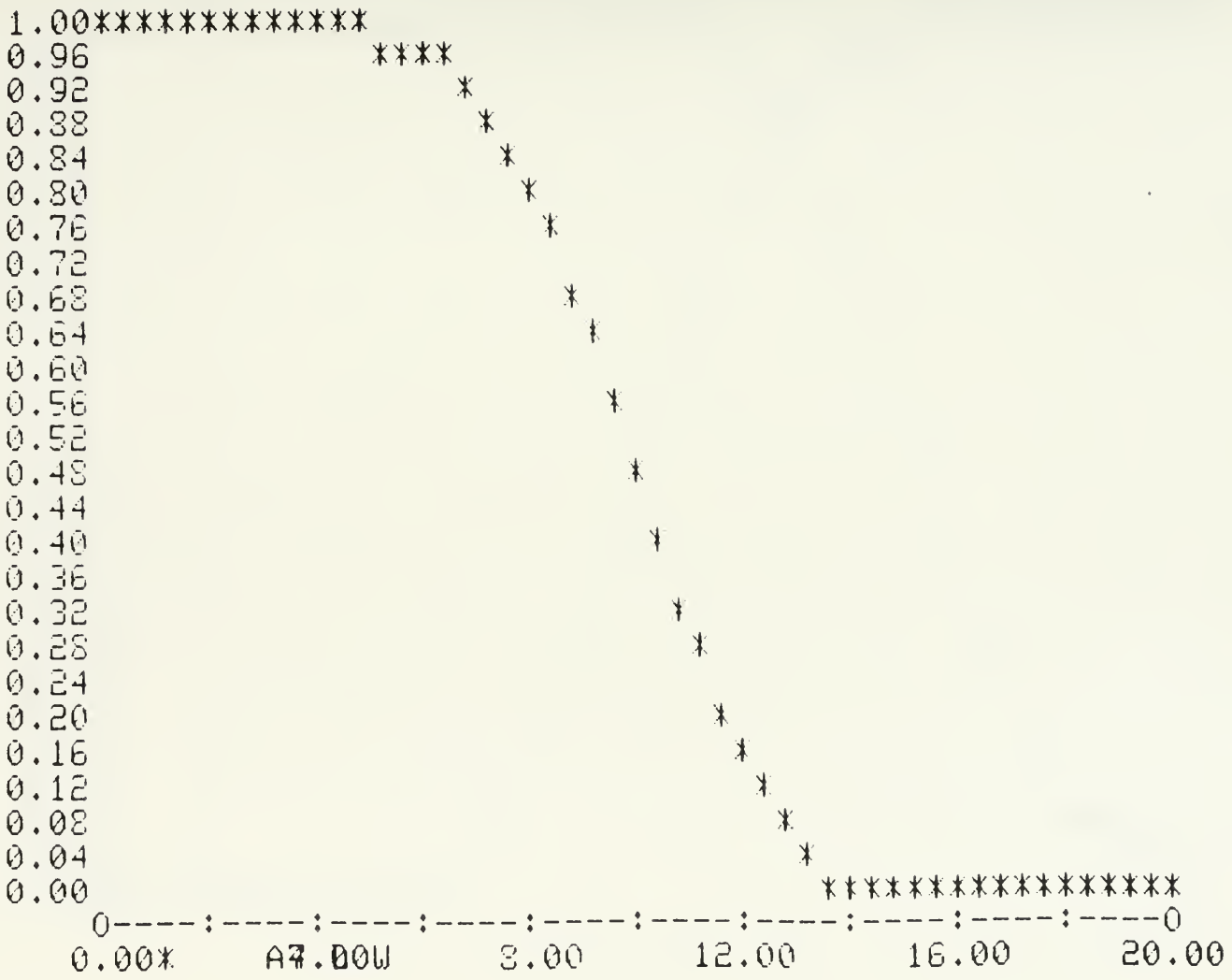

```
MODE >
```

```
draw a6.decreased
```




```
MODE >
```

```
draw a7.low
```

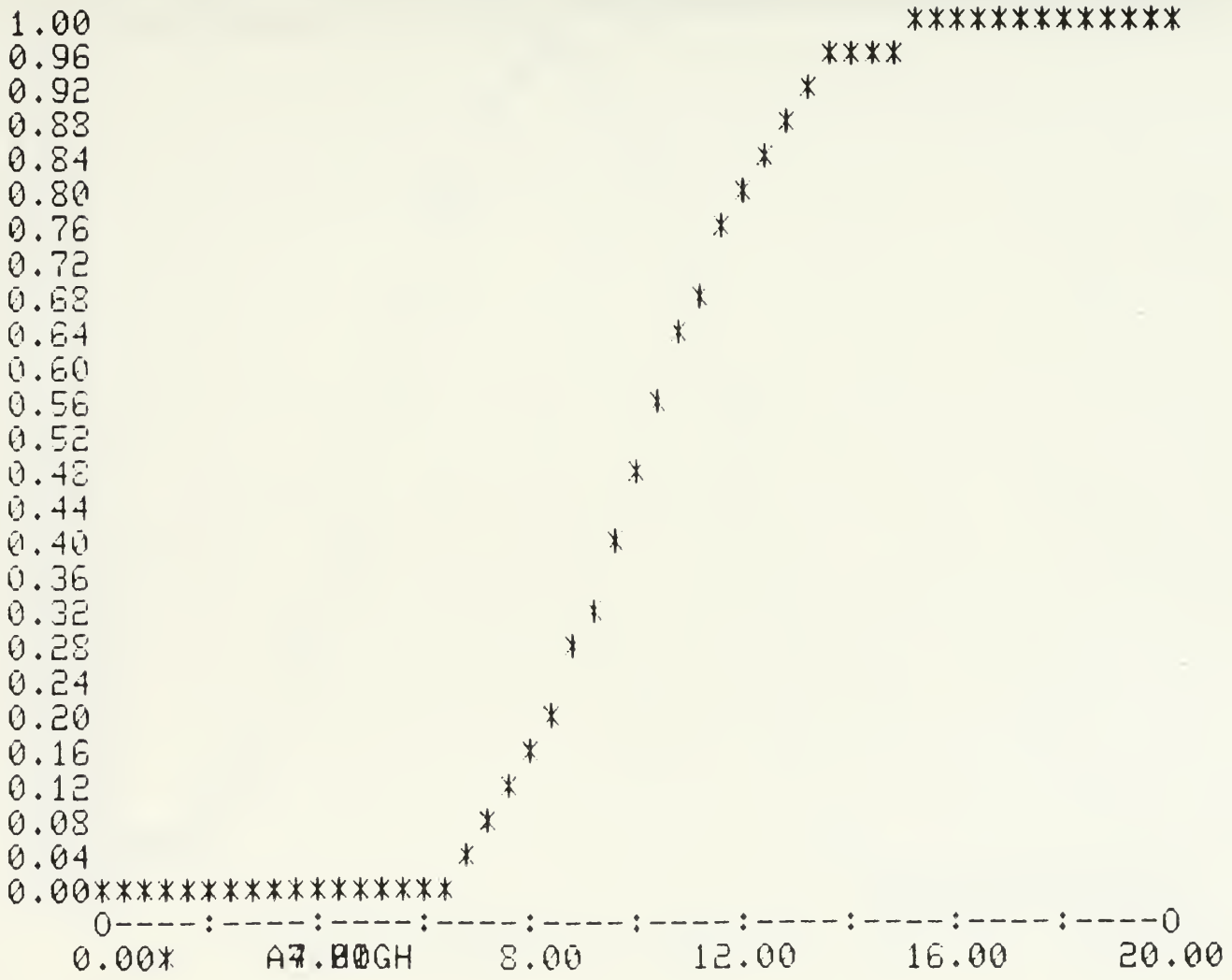



```
MODE >
```

```
draw a7.increased
```

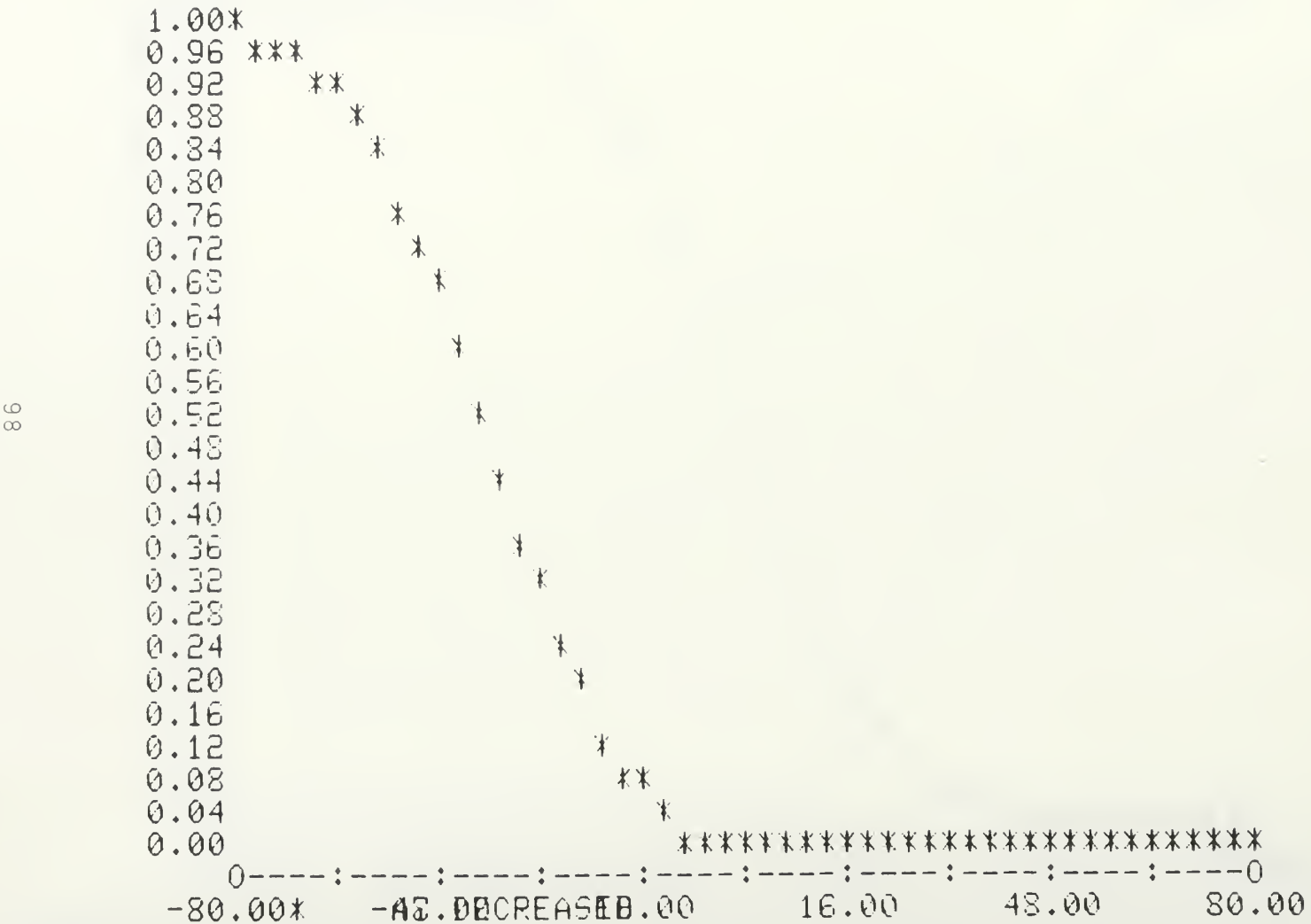

```
MODE >
```

```
draw a7.high
```



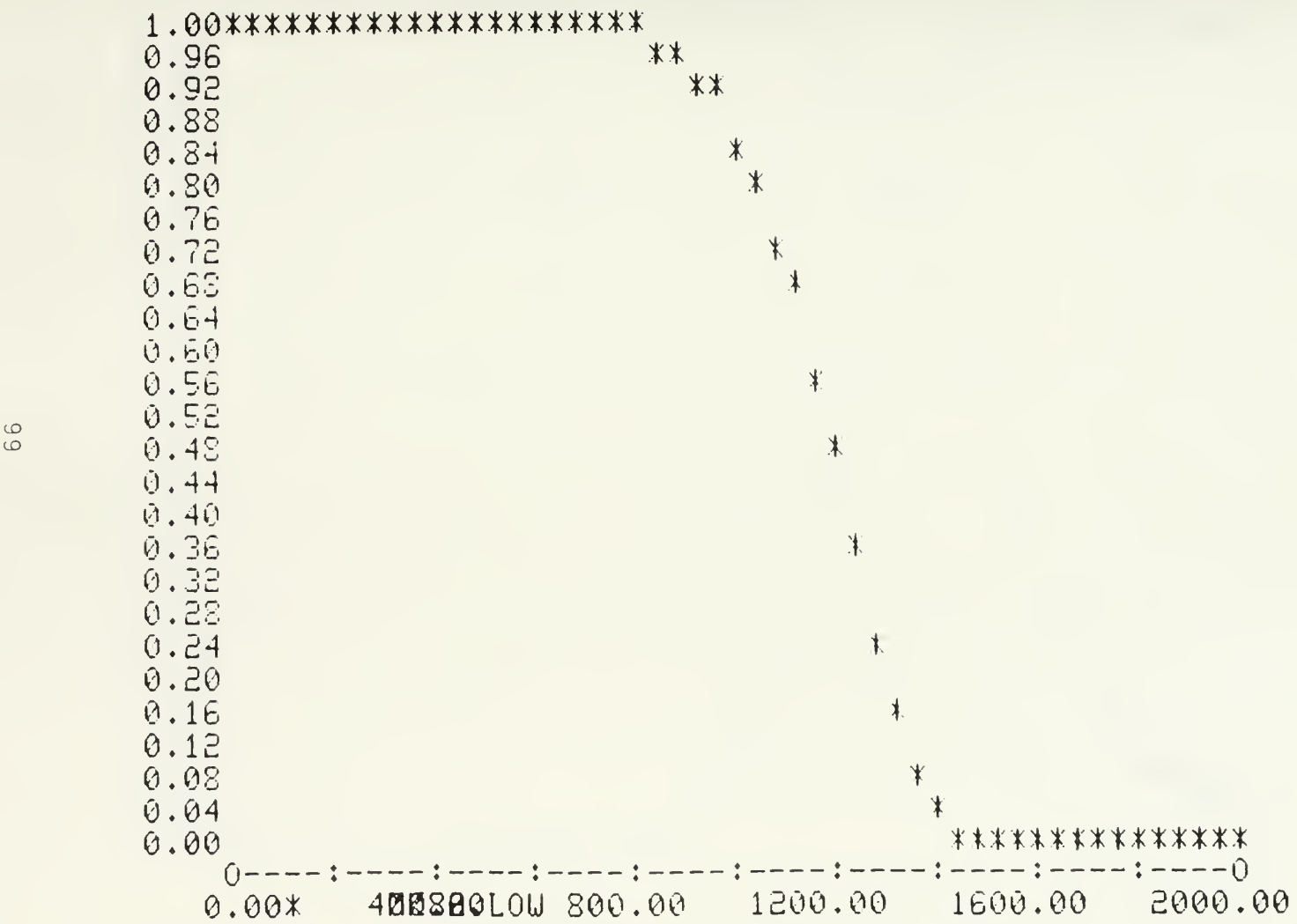

```
MODE >
```

```
draw a7.decreased
```




```
MODE >
```

```
draw mk82.low
```

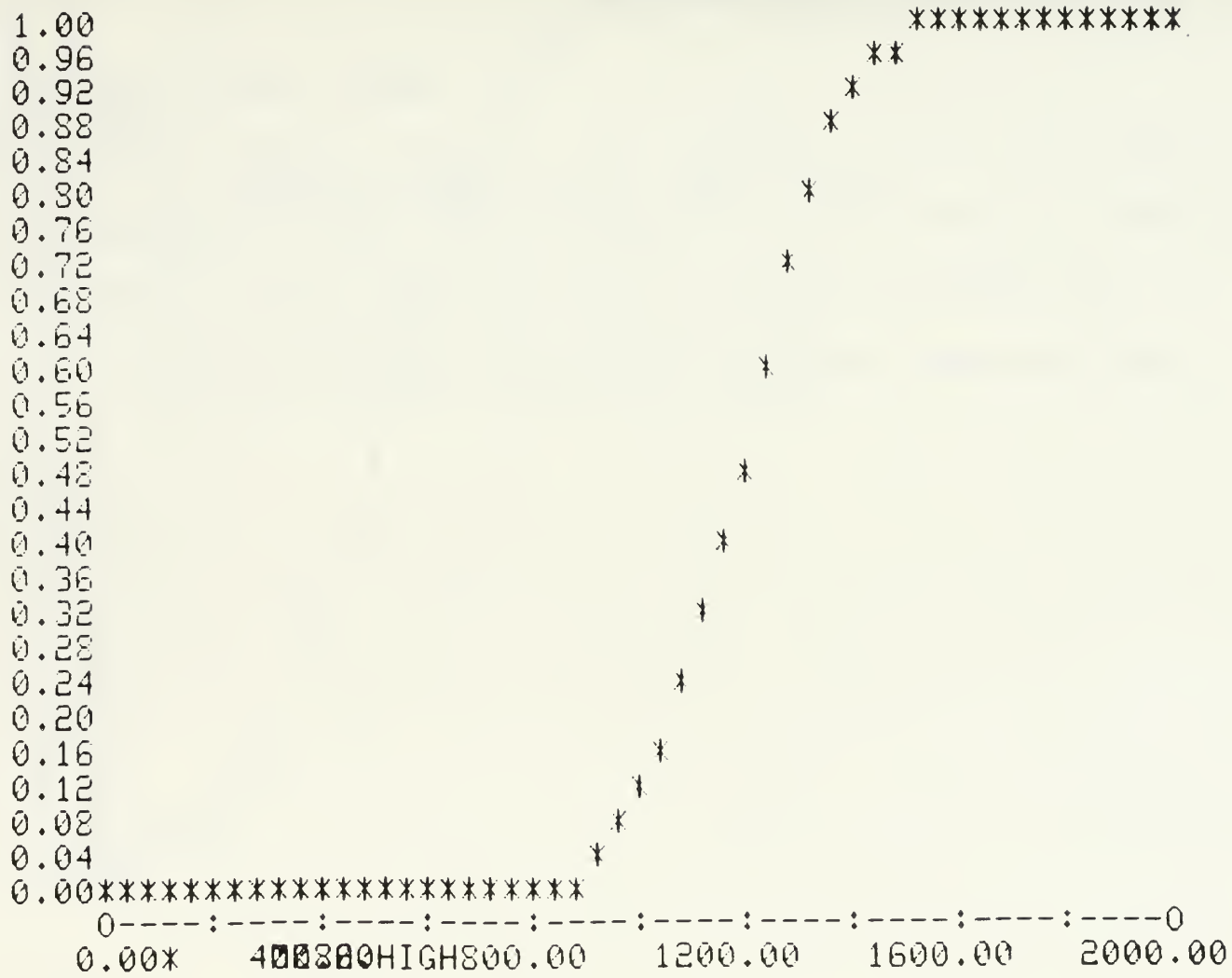



```
MODE >
```

```
draw mk82.incrsd
```

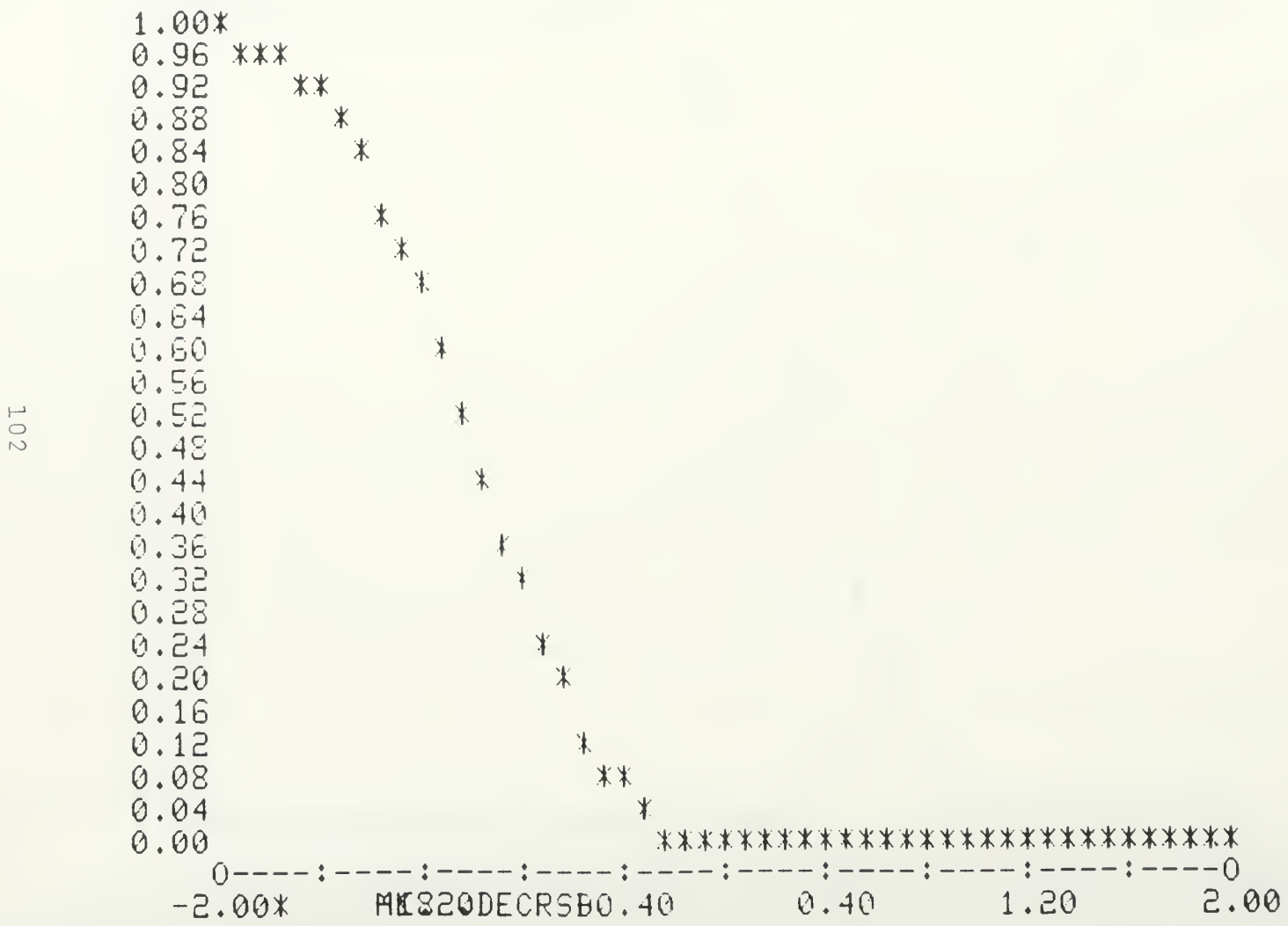

```
MODE >
```

```
draw mk82.high
```




```
MODE >
```

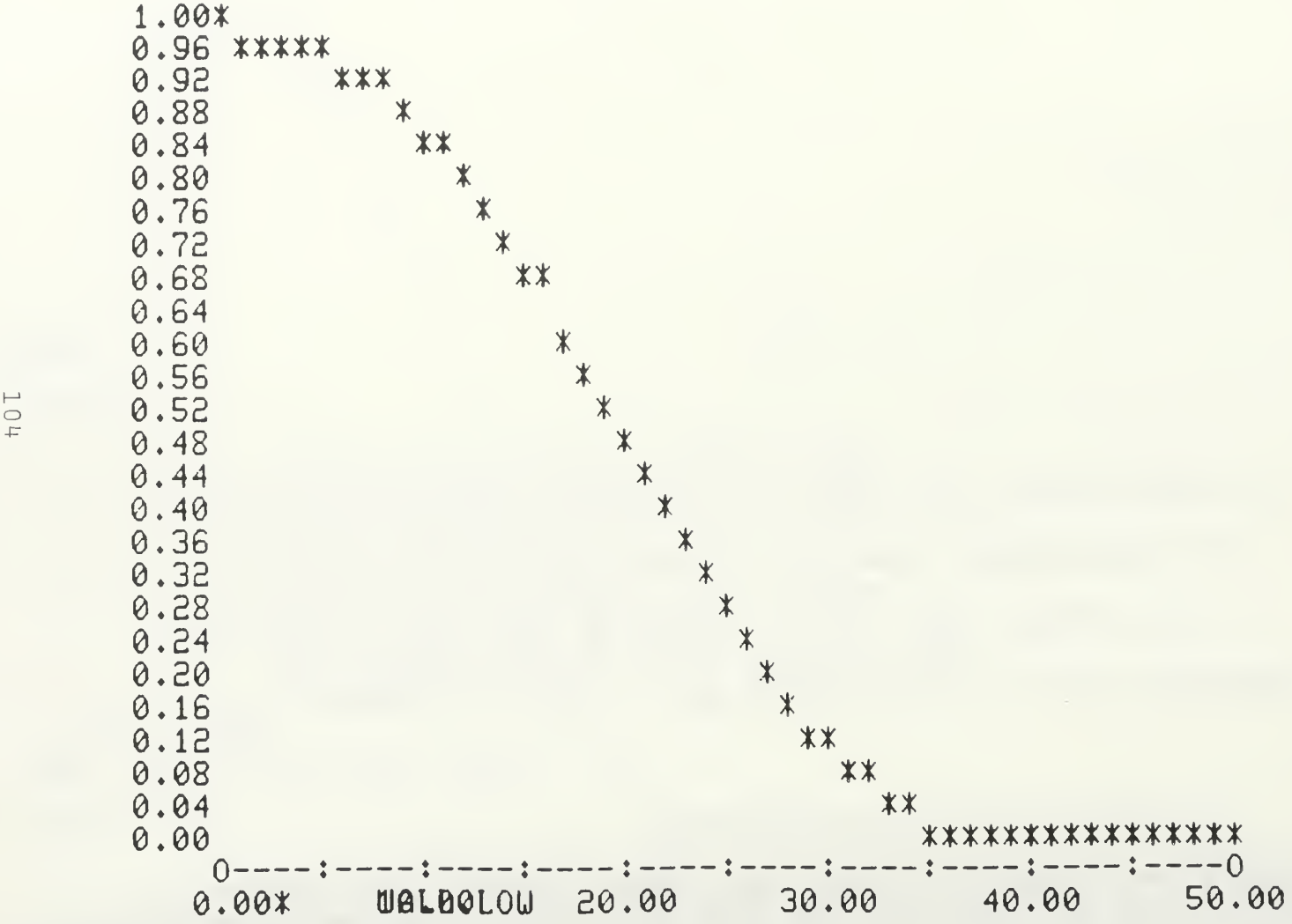
```
draw mk82.decrsd
```




```
MODE >
```

```
policy wall.avail
*>type 1 100
  1 : if walleye.ava is wall.low then wal.val.chng is wall.incrsd
  2 : if walleye.ava is wall.high then wal.val.chng is wall.decrsd

*>quit

MODE >policy a7.wall.load
*>type 1 100
  1 : if forty.degree is true   and distance is far and inbound.thrt
          is strong then a7.wall.load is one
  2 : if pop.up is true and distance is far and inbound.thrt is
          strong then a7.wall.load is one
  3 : if loft is true and distance is far and inbound.thrt is
          strong then a7.wall.load is one
  4 : if loft is true and distance is close and inbound.thrt is
          strong then a7.wall.load is one

*>quit

MODE >
```

```
draw wall.low
```

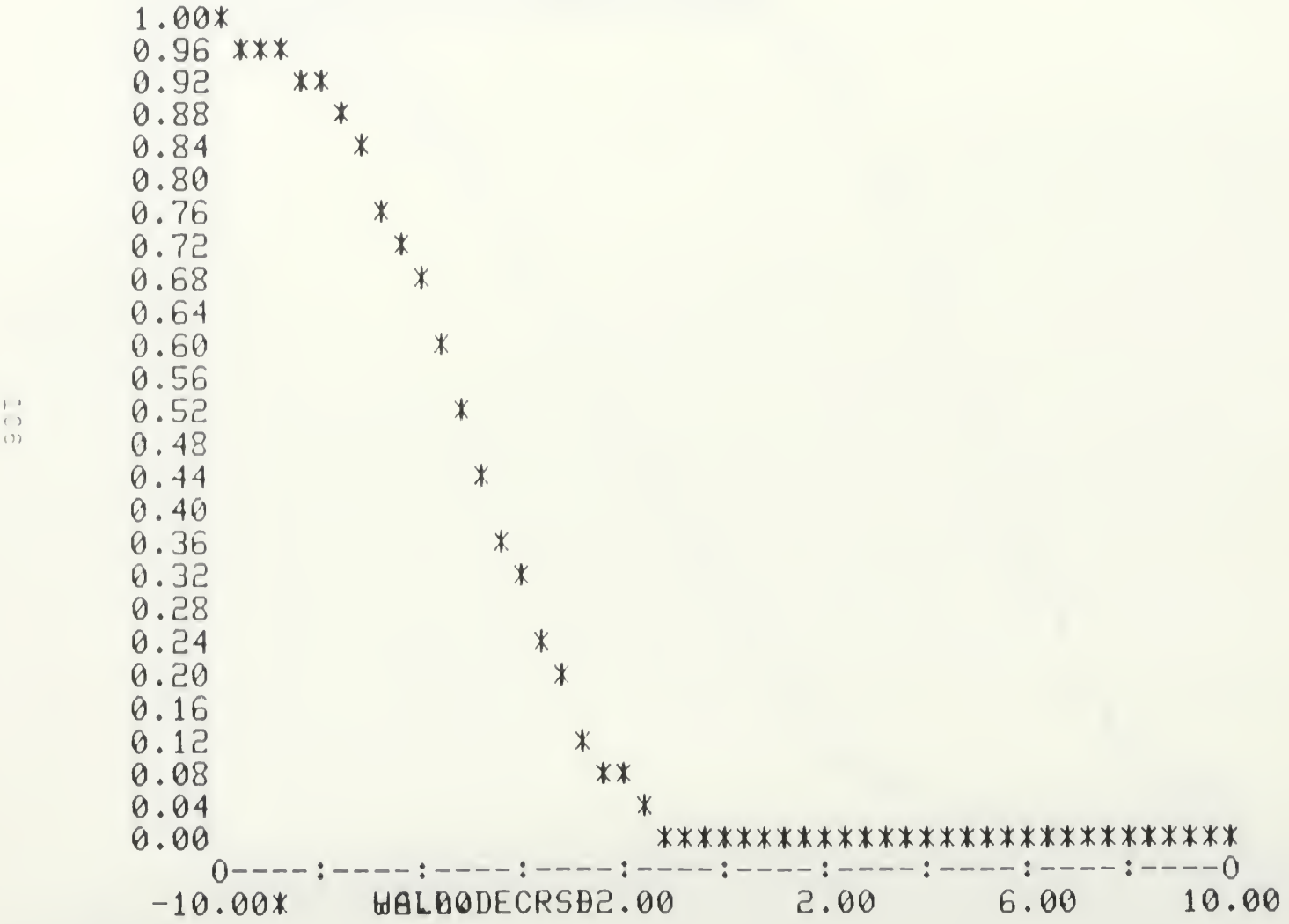



```
MODE >
```

```
draw wall.incrsd
```

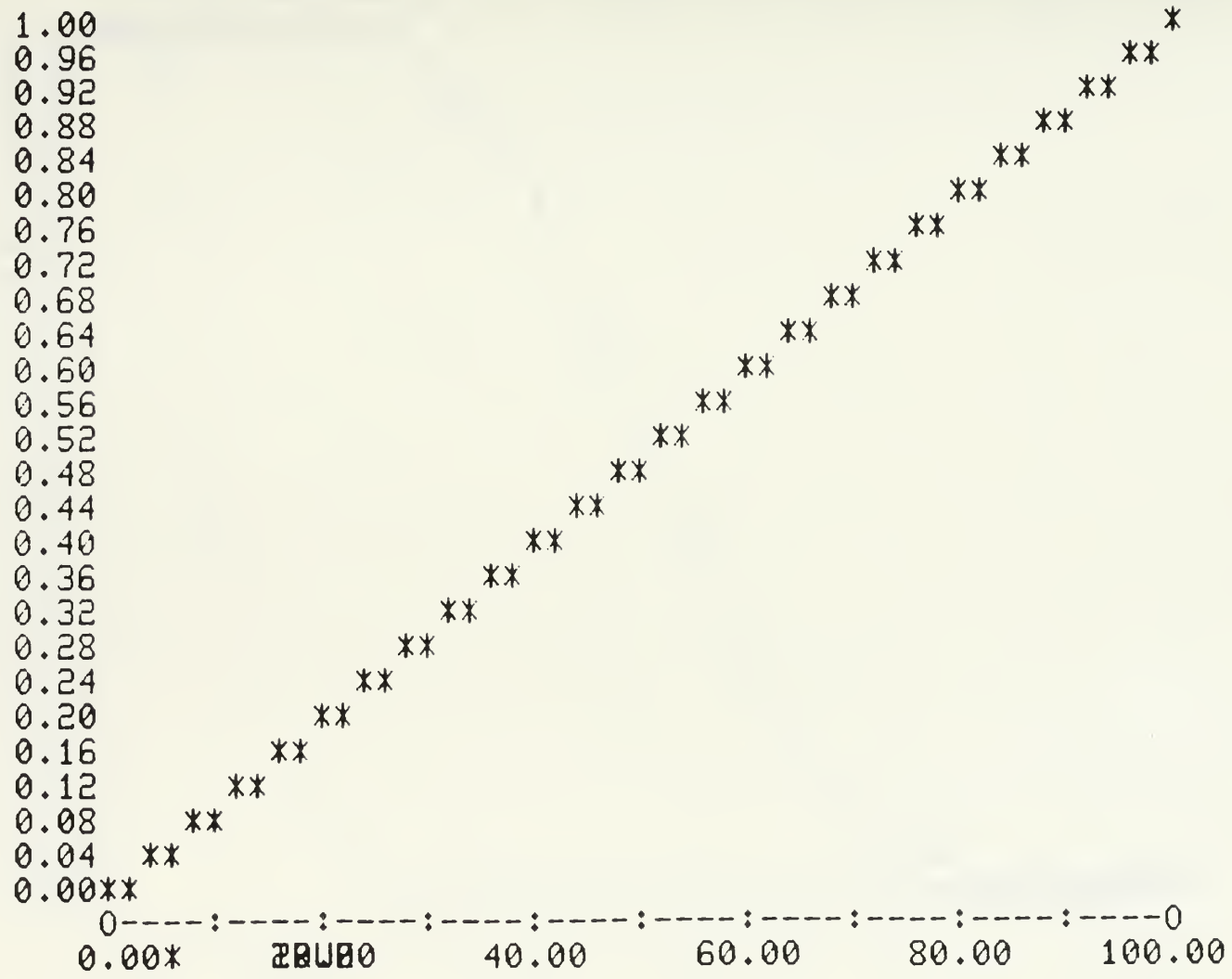

```
MODE >
```

```
draw wall.decrsd
```



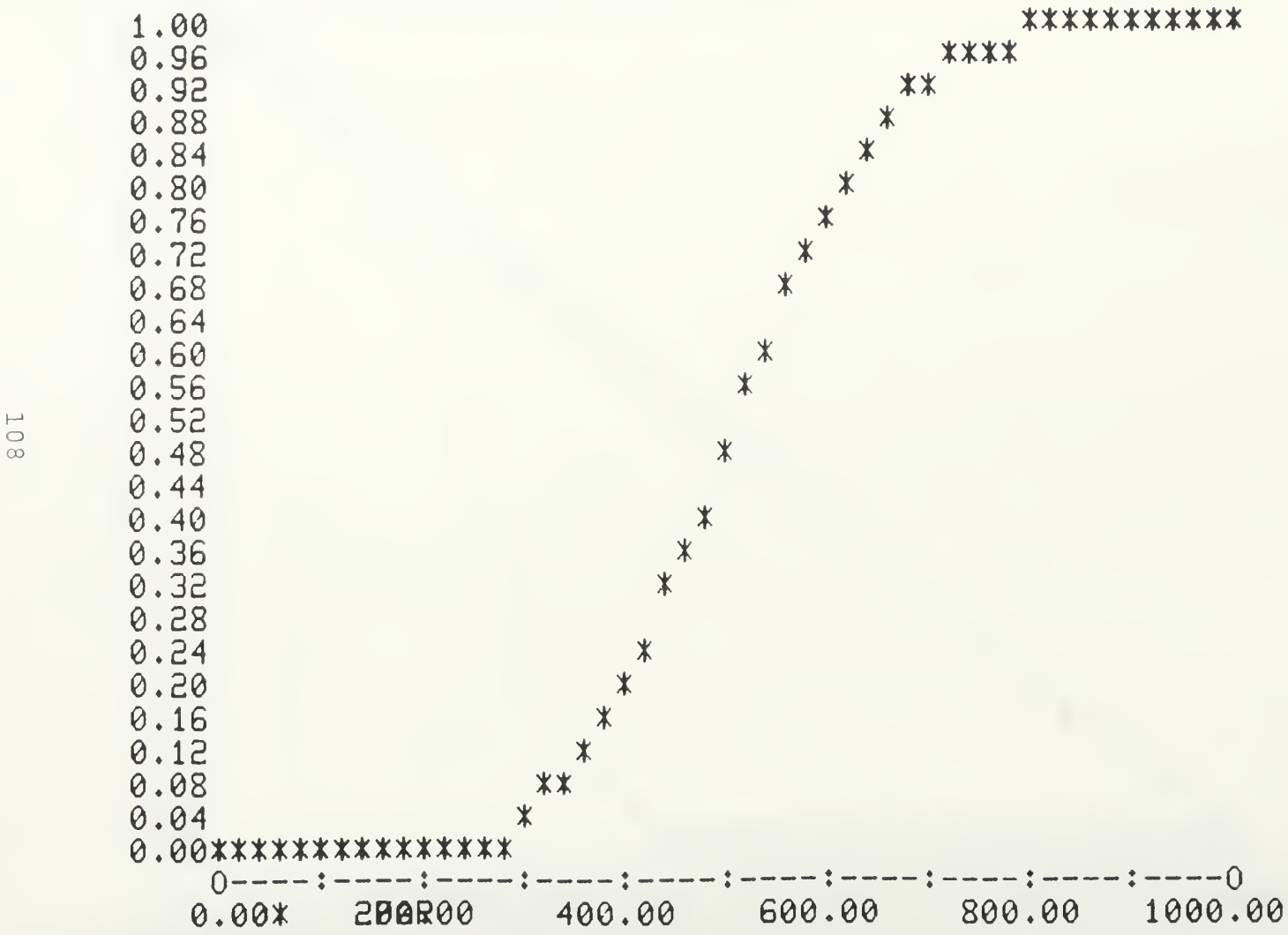

```
MODE >
```

```
draw true
```



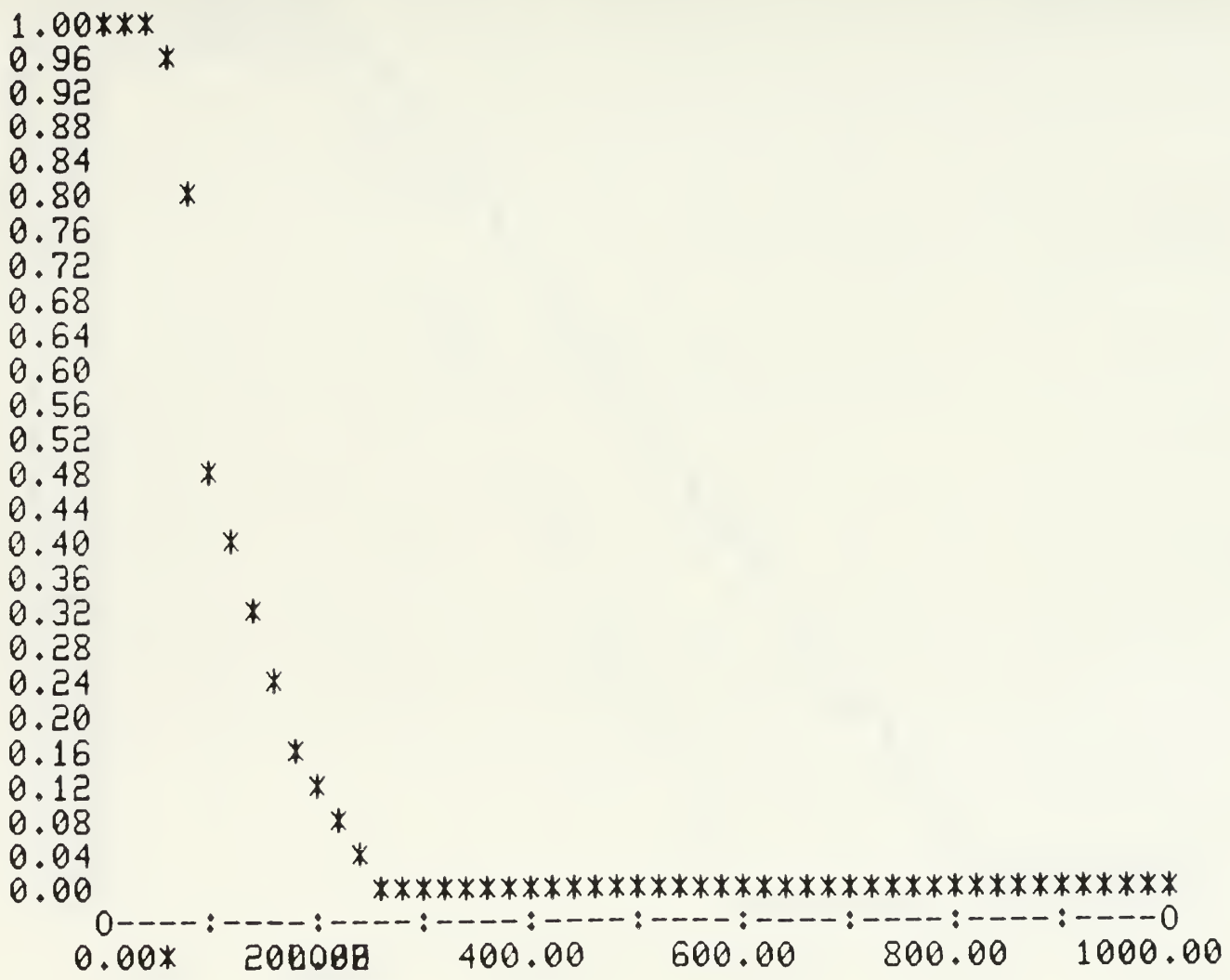

```
MODE >
```

```
draw far
```



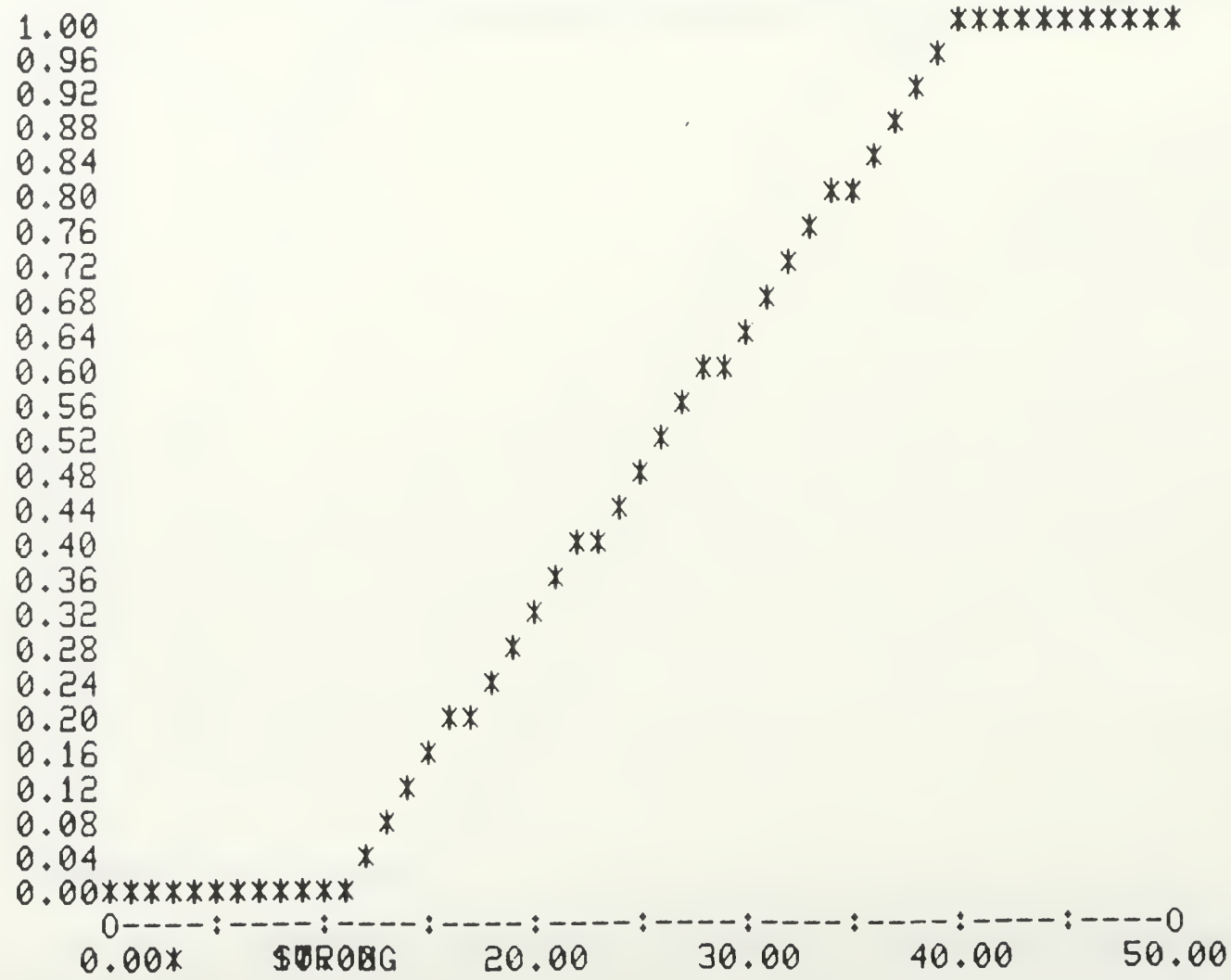

```
MODE >
```

```
draw close
```



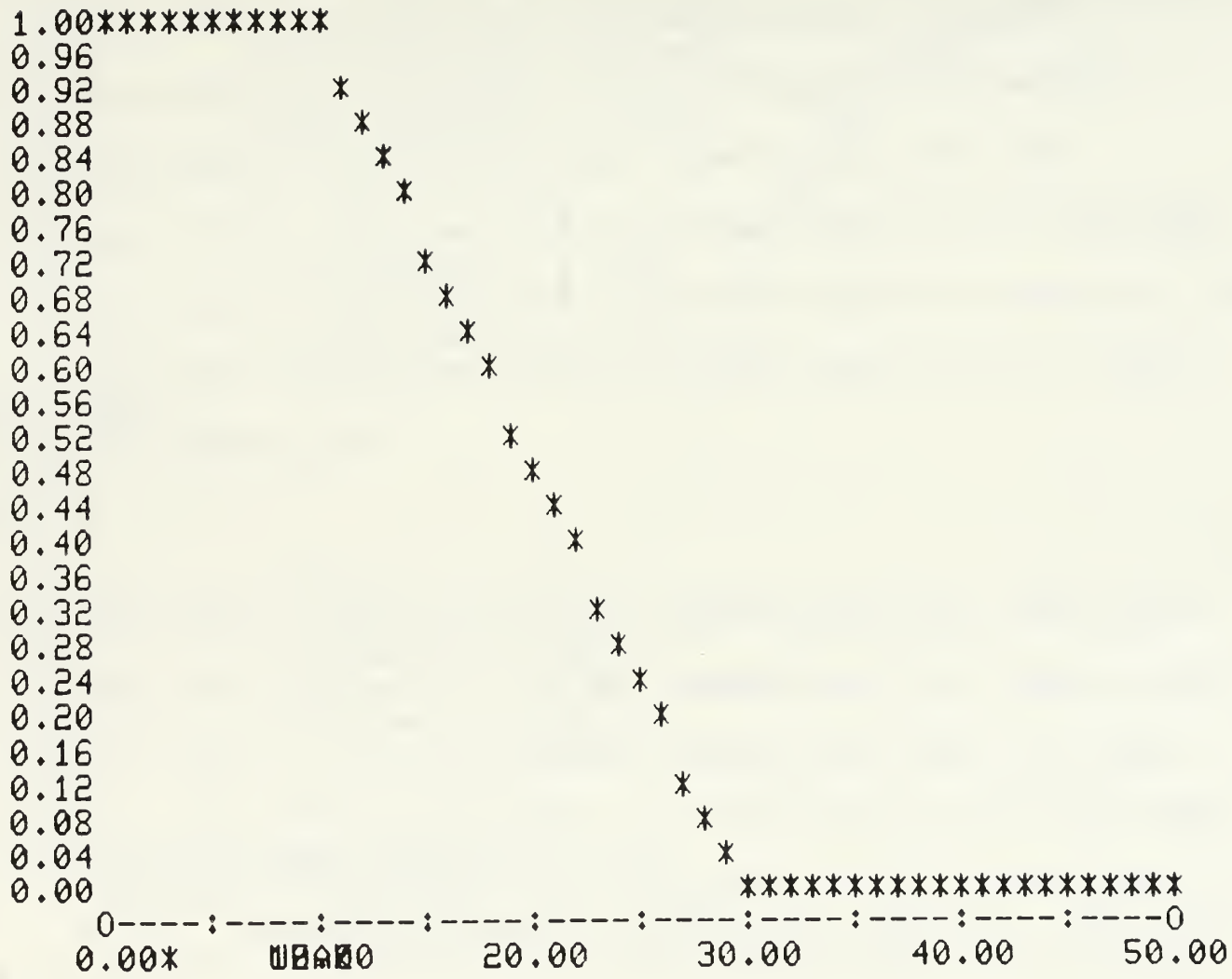

```
MODE >
```

draw strong



MODE >

draw weak



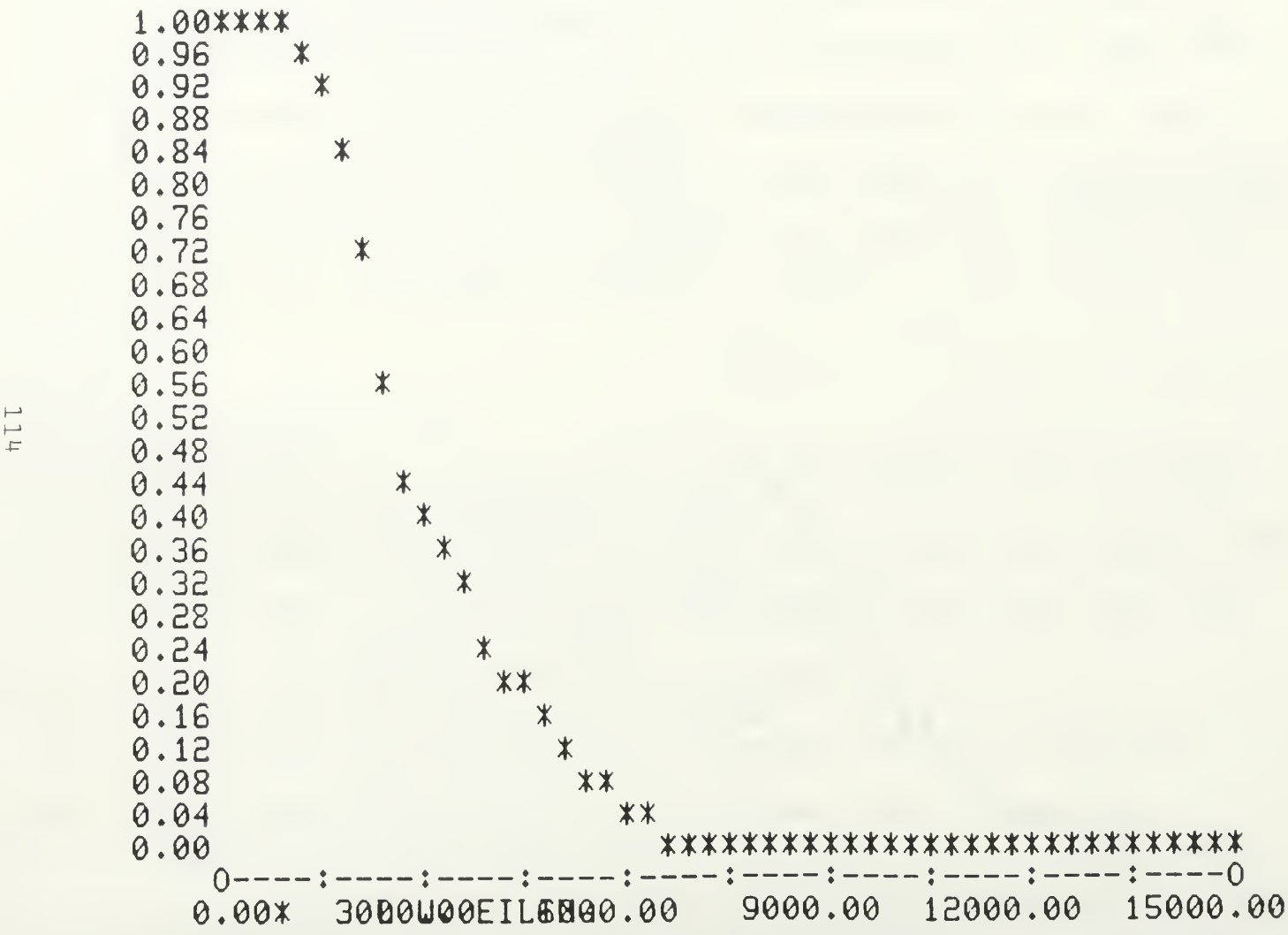
MODE >

```
policy nite.policy
*>type 1 100
  1 : if ceiling is high.ceiling and enemy.batts are low then
                  forty.degree is true
  2 : if ceiling is high.ceiling and enemy.batts are medium
                  then pop.up is true
  3 : if ceiling is high.ceiling and enemy.batts are high then
                  loft is true
  4 : if ceiling is low.ceiling and enemy.batts are low then
                  lay.down is true
  5 : if ceiling is low.ceiling and enemy.batts are medium then
                  lay.down is true
  6 : if ceiling is low.ceiling and enemy.batts are high then
                  lay.down is true

*>quit

MODE >policy day.policy
*>type 1 100
  1 : if ceiling is high.ceiling and enemy.batts are low then
                  forty.degree is true
  2 : if ceiling is high.ceiling and enemy.batts are medium then
                  forty.degree is true
  3 : if ceiling is high.ceiling and enemy.batts are high then
                  pop.up is true
  4 : if ceiling is low.ceiling and enemy.batts are low
                  then lay.down is true
  5 : if ceiling is low.ceiling and enemy.batts are medium
                  then lay.down is true
  6 : if ceiling is low.ceiling and enemy.batts are high
                   then lay.down is true

*>
```

```
draw high.ceiling
```


```
MODE >
```

```
draw low.ceiling
```




```
MODE >
```

draw high

MODE >

draw medium

MODE >

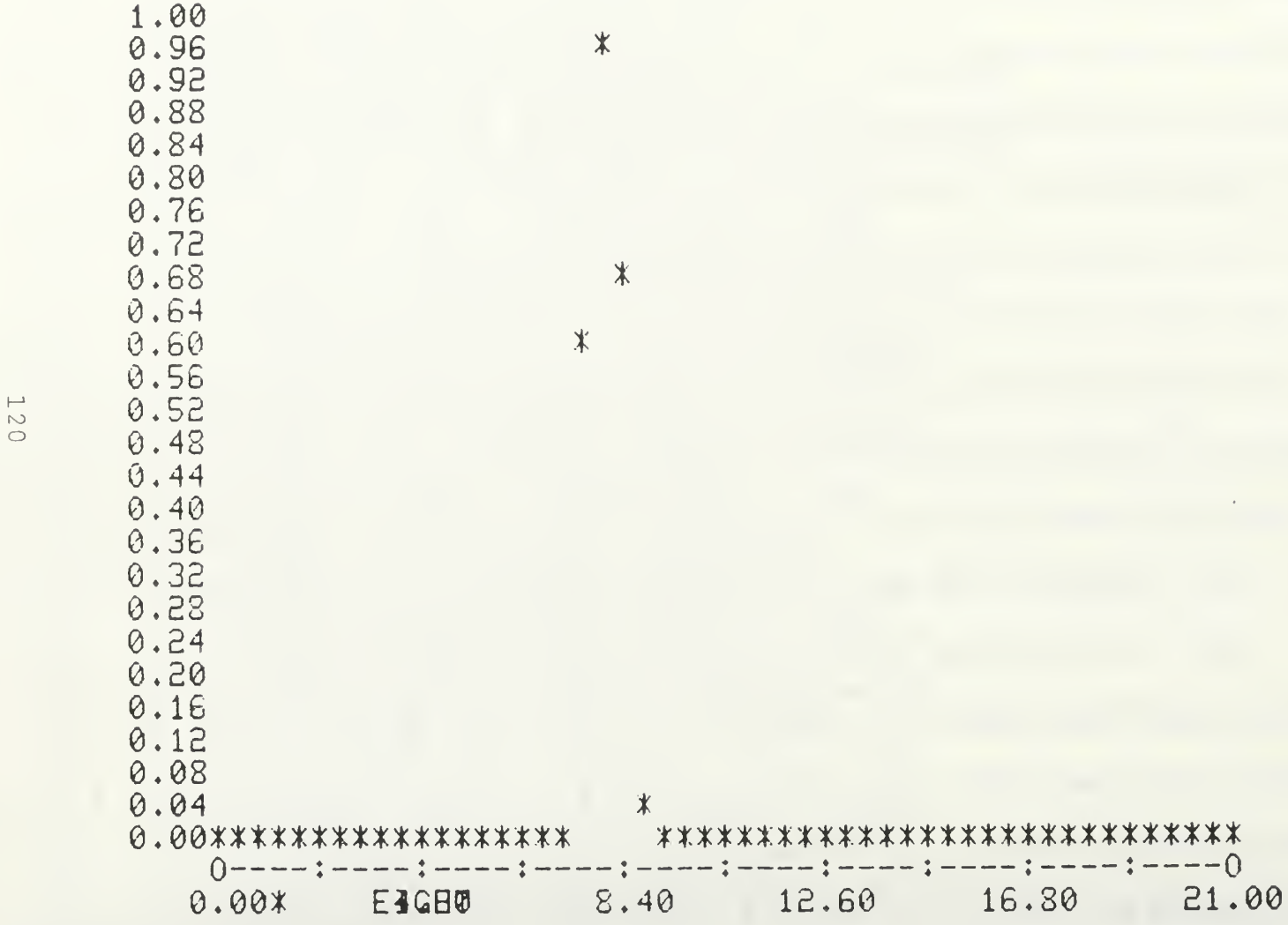
draw low

MODE >

```
olicy a6.mk82.load                    *)
*)type 1 100
  1 : if forty.degree is true and distance is far and inbound.thrt
          is strong then a6.mk82.load is eight
  2 : if forty.degree is true and distance is far and inbound.thrt
          is weak then a6.mk82.load is twelve
  3 : if forty.degree is true and distance is close and inbound.thrt
          is strong then a6.mk82.load is sixteen
  4 : if forty.degree is true and distance is close and inbound.thrt
          is weak then a6.mk82.load is eighteen
  5 : if lay.down is true and distance is far and inbound.thrt is
          strong then a6.mk82.load is ten
  6 : if lay.down is true and distance is far and inbound.thrt is
          strong then a6.mk82.load is sixteen
  7 : if lay.down is true and distance is close and inbound.thrt is
          weak then a6.mk82.load is eighteen
  8 : if lay.down is true and distance is close and inbound.thrt is
          weak then a6.mk82.load is twenty
  9 : if pop.up is true and distance is far and inbound.thrt is strong
          then a6.mk82.load is ten
 10 : if pop.up is true and distance is far and inbound.thrt is weak
          then a6.mk82.load is twelve
 11 : if pop.up is true and distance is close and inbound.thrt is strong
          then a6.mk82.load is twelve
 12 : if pop.up is true and distance is close and inbound.thrt is weak
          then a6.mk82.load is sixteen
 13 : if loft is true and distance is far and inbound.thrt is strong
          then a6.mk82.load is six
 14 : if loft is true and distance is far and inbound.thrt is weak
          then a6.mk82.load is eight
 15 : if loft is true and distance is close and inbound.thrt is strong
          then a6.mk82.load is eight
 16 : if loft is true and distance is close and inbound.thrt is weak
          then a6.mk82.load is ten
```

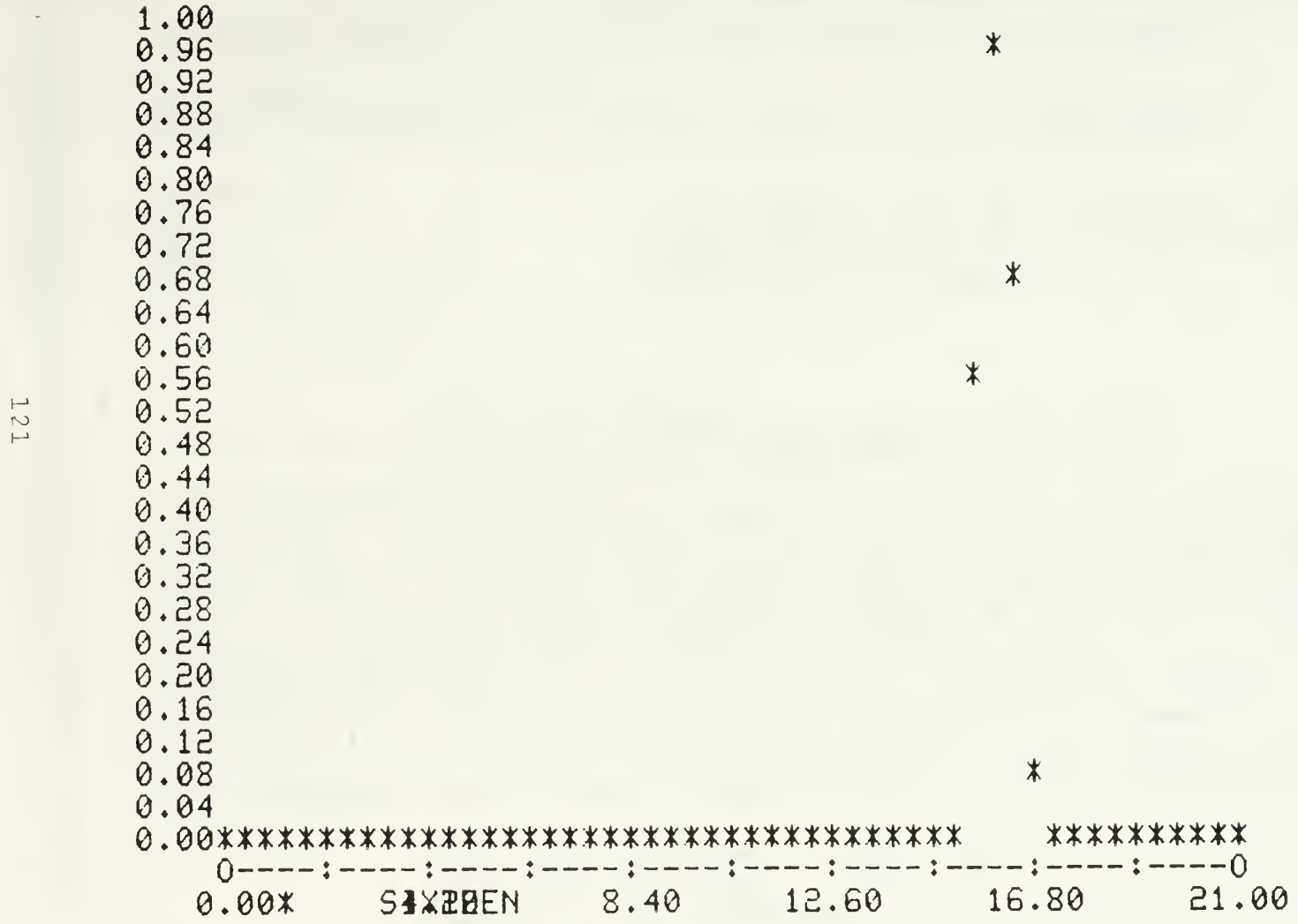
```
policy a7.mk82.load                    *)
*)type 1 100
  1 : if forty.degree is true and distance is far and inbound.thrt
        is strong then a7.mk82.load is six
  2 : if forty.degree is true and distance is far and inbound.thrt
        is weak then a7.mk82.load is eight
  3 : if forty.degree is true and distance is close and inbound.thrt
        is strong then a7.mk82.load is ten
  4 : if forty.degree is true and distance is close and inbound.thrt
        is weak then a7.mk82.load is twelve
  5 : if lay.down is true and distance is far and inbound.thrt is
        strong then a7.mk82.load is eight
  6 : if lay.down is true and distance is far and inbound.thrt is
        weak then a7.mk82.load is ten
  7 : if lay.down is true and distance is close and inbound.thrt is
        strong then a7.mk82.load is ten
  8 : if lay.down is true and distance is close and inbound.thrt is
        weak then  a7.mk82.load is twelve
  9 : if pop.up is true and distance is far and inbound.thrt is
        strong then a7.mk82.load is eight
 10 : if pop.up is true and distance is far and inbound.thrt is
        weak then a7.mk82.load is eight
 11 : if pop.up is true and distance is close and inbound.thrt is
        strong then a7.mk82.load is ten
 12 : if pop.up is true and distance is close and inbound.thrt is
        weak then a7.mk82.load is twelve
 13 : if loft is true and distance is far and inbound.thrt is strong
        then a7.mk82.load is six
 14 : if loft is true and distance is far and inbound.thrt is weak
        then a7.mk82.load is six
 15 : if loft is true and distance is close and inbound.thrt is strong
        then a7.mk82.load is eight
 16 : if loft is true and distance is close and inbound.thrt is weak
        then a7.mk82.load is ten
```

draw eight



MODE >

```
draw sixteen
```




```
MODE >
```

```
type 1 30
  1 : !                Title: Tutor
  2 : !                Author: D.W. Verhagen
  3 : !                Date: 10 JAN 1985
  4 : !                Date Last Update: 14 MAR 1985
  5 : !
  6 : !    Summary: This is the command module for an online REVEAL
  7 : ! tutorial.  This command module loads a context and runs a
  8 : ! program that requests user input responses to a menu.  The
  9 : ! program then branches to the area of inquiry and sample
 10 : ! REVEAL areas are run through the command file.  All vocab-
 11 : ! ularies that are installed are cleaned up through deletion.
 12 : ! The context Tutorsup was required due to the limited REVEAL
 13 : ! character string storage. To run the tutorial simply logon
 14 : ! and enter your REVEAL workarea. The tutorial can be accessed
 15 : ! by entering Doug:tutor
 16 : !                Context: Tutor, Tutorsup
 17 : !                Vocabulary: None
 18 : !                Policies: None
 19 : !
 20 : !
 21 : load doug:tutor          !Loads the main menu
 22 : execute t1 t1            !executes (initially and when called)
 23 : load doug:tutorsup      !Loads when called by VOCABULARY (6)
 24 : execute t1 t1
 25 : !
 26 : !Sample vocabulary is installed and reviewed and cleanup is
 27 : !performed by deleting.
 28 : !
 29 : vocabulary tutor
 30 : y                        ! Response to new vocabulary

*>
```

```
type 30 60
 30 : y                              ! Response to new vocabulary
 31 : dictionary
 32 : delete vocabulary tutor
 33 : execute t1 t1
 34 : !
 35 : !Sample qualifier is created and cleanup is performed
 36 : !
 37 : vocabulary tutor
 38 : y                              ! Response to new vocabulary?
 39 : define long
 40 : 100                            ! low limit
 41 : 1000                           ! high limit
 42 : grow(100,500,1000)             ! membership function
 43 : draw long
 44 : delete vocabulary tutor
 45 : execute t1 t1
 46 : !
 47 : !Sample qualifier is produced and cleanup performed
 48 : !
 49 : vocabulary tutor
 50 : y
 51 : define long
 52 : 100
 53 : 1000
 54 : grow(100,500,1000)
 55 : produce short
 56 : not long
 57 : draw long short
 58 : delete vocabulary tutor
 59 : execute t1 t1
 60 : !

*)
```

```
type 60 90
 60 : !
 61 : !Sample hedge is created, displayed and cleanup is performed.
 62 : !
 63 : vocabulary tutor
 64 : y
 65 : define long
 66 : 100
 67 : 1000
 68 : grow(100,500,1000)
 69 : hedge somewhat .5
 70 : draw long somewhat long
 71 : delete vocabulary tutor
 72 : execute t1 t1

*)
```

```
type 1 30
  1 : ! Title: Tutor
  2 : ! Author: D.W. Verhagen
  3 : ! Date 1 FEB 1985
  4 : ! Date Last Updated: 14 MAR 1985
  5 : !
  6 : !Summary: This program is called by the tutor command file and
  7 : !presents a menu to the user from which he can make inquiries
  8 : !into various REVEAL areas.   This file returns control to
  9 : !the command file. If item six, VOCABULARY, is selected the
 10 : !control is passed to the tutorsup file via the command file.
 11 : !
 12 : !
 13 : print (' ')
 14 : print ('Hit enter to continue')
 15 : continue = tty
 16 : begin:
 17 : !
 18 : print ('WELCOME to the REVEAL tutorial. This tutorial is ')
 19 : print ('designed to give you a brief overview of the REVEAL ')
 20 : print ('software system and how it is used.  Some modules con- ')
 21 : print ('tain stubs so if your looking for a thesis topic....')
 22 : print (' ')
 23 : print ('Please select the area you are interested in:  ')
 24 : print (' ')
 25 : print (' ')
 26 : print ('          Exit ----------------0    ')
 27 : print ('          Fuzzy Sets ----------1    ')
 28 : print ('          REVEAL Environment --2    ')
 29 : print ('          User Context --------3     ')
 30 : print ('          Data Entry ----------4    ')

*>
```

```
type 31 59
 31 : print ('          Policies ------------5     ')
 32 : print ('          Vocabulary ----------6     ')
 33 : print ('          Programming ---------7     ')
 34 : !
 35 : flag = tty
 36 : !
 37 : if flag EQ 0 then do
 38 :      branch 9999
 39 :   enddo
 40 : !
 41 : !This section talks about Fuzzy Sets
 42 : !
 43 : if flag EQ 1 then do
 44 : print ('FUZZY SETS')
 45 : print (' ')
 46 : print ('   Fuzzy set theory was proposed by Lotfi Zadeh in 1965. ')
 47 : print ('A fuzzy set is a class where there is a continuum of mem- ')

 48 : print ('bership.  The set of Tall men or Small girls are fuzzy sets.
 ')
 49 : print ('Fuzzy sets play an important role in human cognition and ')
 50 : print ('underlie our ability to make effective decisions with ')
 51 : print ('partial information.  The notions of union, intersection, ')

 52 : print ('complement, etc. are extended to these sets. Linguistic ')
 53 : print ('as well as numeric variables can be acted on. ')
 54 : print (' ')
 55 : print ('Hit enter to continue')
 56 : continue = tty
 57 : branch 21
 58 : enddo
 59 : !

*)
```

```
type 60 85
 60 : !This section talks about the REVEAL environment
 61 : !
 62 : If flag EQ 2 then do
 63 : print ('REVEAL ENVIRONMENT')
 64 : print (' ')
 65 : print ('      This module is currently a stub    ')
 66 : !
 67 : print (' ')
 68 : print ('Hit enter to continue')
 69 : continue = tty
 70 : branch 21
 71 : enddo
 72 : !
 73 : !
 74 : !This section talks about the REVEAL user context
 75 : !
 76 : If flag EQ 3 then do
 77 : print ('USER CONTEXT')
 78 : print (' ')
 79 : print ('   Logging onto the computer and entering REVEAL puts ')
 80 : print ('the user into the following context:    ')
 81 : print (' ')
 82 : print (' Data matrices ')
 83 : print ('   1. A two dimensional matrix of 1000 series and 200 terms
')
 84 : print ('   2. A one dimensional matrix of 1000 constants   ')
 85 : print ('   3. A one dimensional matrix of 1000 char string variables
')

*)
```

```
type 85 115
 85 : print ('a decreasing s-shaped, or a bell curve.  The following ')
 86 : print ('formats apply: ')
 87 : print ('       Decline(true,mid,false): declining s-shaped  ')
 88 : print ('       Grow(false,mid,true): increasing s-shaped    ')
 89 : print ('       Line(false,true): increasing/decreasing line ')
 90 : print ('       Pi(true,midwidth): bell-shaped curve         ')
 91 : print ('We will now examine the set membership function for long ')
 92 : print ('using a low limit of 100 ft, a high limit of 1000 ft and ')
 93 : print ('a set membership function of GROW(100,500,1000). ')
 94 : print (' ')
 95 : print (' ')
 96 : print ('Hit enter to continue.')
 97 : continue = tty
 98 : branch 37
 99 : enddo
100 : !
101 : ! This section discusses the PRODUCE method of creating a qualifier.

102 : ! An example is run through the tutor command module and we're re-
103 : ! turned to the menu of this module.
104 : !
105 : if flag eq 3 then do
106 : print ('QUALIFIERS ---PRODUCE')
107 : print (' ')
108 : print ('   Qualifiers may also be produced as opposed to ')
109 : print ('defined.  This is done when the qualifier to be ')
110 : print ('produced bears some relation to an already defined')
111 : print ('qualifier.  For instance, short may be defined as ')
112 : print ('not long.  The command PRODUCE <qualifier name> is ')
113 : print ('entered.  REVEAL asks for the function and, in this ')
114 : print ('case, not long would be entered.  We will now examine')
115 : print ('the qualifier-short produced from the qualifier-long ')

*>
```

```
type 115 140
115 : print ('the qualifier-short produced from the qualifier-long ')
116 : print ('by defining it as not long: ')
117 : print (' ')
118 : print (' ')
119 : print ('Hit enter to continue.')
120 : continue = tty
121 : branch 49
122 : enddo
123 : !
124 : ! This section discusses hedges.  It runs a sample hedge through the

125 : ! the tutor command file and returns to the menu in this module.
126 : !
127 : if flag eq 4 then do
128 : print ('HEDGES')
129 : print (' ')
130 : print ('     Hedges act to amplify or decrease the impact of a ')
131 : print ('qualifier.  Examples of hedges are very, quite, not, ')
132 : print ('more than and about.  REVEAL contains certain default ')
133 : print ('hedges and they can be reviewed by entering DICTIONARY ')
134 : print ('HEDGE. If a hedge needs to be created in can be done by ')
135 : print ('entering HEDGE <hedge name> <value>.  We will now examine ')

136 : print ('a hedge--somewhat with a value of .5.  The graph is of ')
137 : print ('long and somewhat long. ')
138 : print (' ')
139 : print (' ')
140 : print ('Hit enter to continue.')

*>
```

```
type 145 175
145 : !
146 : !this section talks about programming
147 : !
148 : If flag EQ 7 then do
149 : print ('REVEAL PROGRAMMING')
150 : print (' ')
151 : print ('    Once vocabularies are installed, constants are ')
152 : print ('named, and policies established programming can begin. ')
153 : print ('The program ties all the REVEAL features together.  The ')
154 : print ('program is entered by typing EDIT. This puts the user in ')
155 : print ('the edit mode and program entry can begin.  To save a ')
156 : print ('program the STORE <filename> command is used and to re- ')
157 : print ('trieve a program the LOAD command is used.  REVEAL uses ')
158 : print ('IF-THEN-ELSE statements, GOTO statements, REPEAT statements,
 ')
159 : print ('and has many unique and useful built-in functions. The user
')
160 : print ('is refered to the REVEAL Manual, chapter 5 for further ')
161 : print ('discussion')
162 : print (' ')
163 : print ('Hit enter to continue')
164 : branch 21
165 : enddo
166 : If Flag GT 7 then goto begin:

*>
```

```
type 1 30
  1 : ! Title: Tutorsup, (Supplemental for VOCABULARY module)
  2 : ! Author: D. W. Verhagen
  3 : ! Date: 18 FEB 1985
  4 : ! Date Last Updated: 13 MAR 1985
  5 : !
  6 : ! This module is referred to by the tutor module and
  7 : ! depending on input refers to the appropriate section
  8 : ! to answer user inquiries.  All referal action is done
  9 : ! by the tutor command module and this module returns
 10 : ! control to the tutor command module.
 11 : !
 12 : !
 13 : ! The continue variable entry prevents scrolling of the page
 14 : ! which prevents reading-- unless you are a speed reader
 15 : !
 16 : print (' ')
 17 : print ('Hit enter to continue.')
 18 : continue = tty
 19 : !
 20 : ! This section asks about specific vocabulary areas. A flag is then
 21 : ! set and the appropriate section called using IF-THEN statements.
 22 : !
 23 : print ('REVEAL VOCABULARIES')
 24 : print (' ')
 25 : print ('    REVEAL vocabularies are powerful tools that when')
 26 : print ('coupled with policies makes REVEAL an excellent model-')
 27 : print ('ling language for complex, real world problems.  Infor-')
 28 : print ('mation is available in this tutorial in the following ')
 29 : print ('areas:  ')
 30 : print (' ')

*)
```

```
type 30 60
 30 : print (' ')
 31 : print (' ')
 32 : print ('                    Return to Main Menu --------- 0    ')
 33 : print ('                    Installing a Vocabulary ----- 1    ')
 34 : print ('                    Qualifiers  ')
 35 : print ('                        Define ------------------ 2    ')
 36 : print ('                        Produce ----------------- 3    ')
 37 : print ('                    Hedges ---------------------- 4    ')
 38 : print ('                    Noise Words ----------------- 5    ')
 39 : print (' ')
 40 : print ('Make your selection here: ')
 41 : flag = tty
 42 : !
 43 : ! This section returns to the main menu
 44 : !
 45 : if flag eq 0 then do
 46 :     branch 21
 47 :   enddo
 48 : !
 49 : ! This section talks about installing a vocabulary. Through the
 50 : ! tutor command module an example is run and then we're returned
 51 : ! to the menu for this module
 52 : !
 53 : if flag eq 1 then do
 54 : print ('INSTALLING A VOCABULARY')
 55 : print (' ')
 56 : print ('     The default vocabulary contains a few qualifiers, ')
 57 : print ('hedges and noise words. A vocabulary is installed by  ')
 58 : print ('entering VOCABULARY <vocabulary name>.  The words that ')
 59 : print ('are currently in this vocabulary can be reviewed by enter-')

 60 : print ('ing DICTIONARY.  One must be ready to hit the NO SCROLL ')

*>
```

```
type 60 85
 60 : print ('ing DICTIONARY.  One must be ready to hit the NO SCROLL ')
 61 : print ('key as the words stream by rather quickly.  We will now ')
 62 : print ('review a default vocabulary.  Find the NO SCROLL key and')
 63 : print (' ready. ')
 64 : print (' ')
 65 : print (' ')
 66 : print ('Hit enter to continue.')
 67 : continue = tty
 68 : branch 29
 69 : enddo
 70 : !
 71 : ! This section discusses the DEFINE method of creating a qualifier
 72 : ! and runs an example through the tutor command module.  We're
 73 : ! then returned to the menu in this module.
 74 : !
 75 : if flag eq 2 then do
 76 : print ('QUALIFIERS --- DEFINE')
 77 : print (' ')
 78 : print ('     A qualifier can be created by entering DEFINE <qual-')
 79 : print ('ifier name>.  REVEAL will ask for a high limit for the do-')

 80 : print ('main and a low limit for the domain.  It will also ask for '
)
 81 : print ('a function to be used in determining the set membership. ')
 82 : print ('Predefined set membership functions exist or the user may ')

 83 : print ('define their own function.  Predefined functions exist for '
)
 84 : print ('an increasing or decreasing line, an increasing s-shaped ')
 85 : print ('a decreasing s-shaped, or a bell curve.  The following ')

*>
```

```
type 85 115
 85 : print ('   3. A one dimensional matrix of 1000 char string variables
')
 86 : print (' ')
 87 : print (' Data Dictionary ')
 88 : print ('   1. Series (row) indentifiers (names)   ')
 89 : print ('   2. Term (column) identifiers (headings)       ')
 90 : print ('   3. Constant identifiers    ')
 91 : print ('  ')
 92 : print (' Control Parameters and Logic')
 93 : print (' ')
 94 : print ('Hit enter to continue ')
 95 : continue = tty
 96 : branch 21
 97 : enddo
 98 : !
 99 : !
100 : !This section talks about data entry
101 : !
102 : If flag EQ 4 then do
103 : print ('DATA ENTRY')
104 : print (' ')
105 : print ('   Data is assigned to each series column by column. ')
106 : print ('The command ENTER <series id> <term one> <last term> ')
107 : print ('is used. Once data is assigned to one series the ')
108 : print ('procedure is repeated.  Data can also be assigned using ')
109 : print ('datasets.  Constants are assigned by typing CONSTANTS.')
110 : print (' ')
111 : print ('Hit enter to continue ')
112 : continue = tty
113 : branch 21
114 : enddo
115 : !

*>
```

```
type 115 145
115 : !
116 : !
117 : !This section talks about Policies
118 : !
119 : If flag EQ 5 then do
120 : print ('POLICIES')
121 : print (' ')
122 : print ('  The real power of REVEAL lies in its policy analysis.')
123 : print ('Policies are of the form IF <variable> is <fuzzy> THEN ')
124 : print ('<variable> is <fuzzy>. The following are sample policies. ')

125 : print (' ')
126 : print ('If endurance is long or criticallity is essential ')
127 : print ('   then hi.lim.chng is increased          ')
128 : print ('If endurance is normal then hi.lim.chng is nothing ')
129 : print ('If endurance is short and criticallity is nonessential ')
130 : print ('  then hi.lim.chng is decreased          ')
131 : print (' ')
132 : print ('Hit enter to continue ')
133 : continue = tty
134 : branch 21
135 :   enddo
136 : !
137 : !
138 : !This section transfers control to the tutorsup module
139 : !    via the command module
140 : !
141 : If flag EQ 6 then do
142 :    branch 23
143 :   enddo
144 : !
145 : !

*>
```

```
type 140 170
140 : print ('Hit enter to continue.')
141 : continue = tty
142 : branch 63
143 : enddo
144 : !
145 : ! This section discusses noise words and returns to the menu
146 : ! in this section through the tutor command module.  No examples
147 : ! are run.
148 : !
149 : if flag eq 5 then do
150 : print ('NOISE WORDS')
151 : print (' ')
152 : print ('     Words that are not used in a logical relation such ')
153 : print ('as-- if, then, or, and, or as a qualifier, hedge, or var- ')
154 : print ('iable in a REVEAL policy must be defined as a noise word. ')
155 : print ('Noise words are ignored by the compiler when analyzing ')
156 : print ('policy statements.  Their sole function is to make policies'
)
157 : print ('more readable.  Default noise words can be examined by ')
158 : print ('entering DICTIONARY NOISE.  New noise words can be created '
)
159 : print ('by entering NOISE <noiseword>.  ')
160 : print (' ')
161 : print (' ')
162 : print ('Hit enter to continue.')
163 : continue = tty
164 : branch 72
165 : enddo

*>
```

LIST OF REFERENCES

1. Zadeh, L.A., "Fuzzy Sets", Information and Control, v. 8, pp. 338-353, 1965.

2. Zadeh, L.A., "Quantitative Fuzzy Semantics," Information Sciences, v. 3, no. 2, pp. 159-176, April 1971.

3. Janes Publishing Company, Ltd, Janes Fighting Ships 1984-1985, pp. 633-752, 1984.

4. Bellman, R.E. and Zadeh, L.A., "Decision Making in a Fuzzy Environment", Management Science, v. 17, pp. 141-164, 1970.

5. Dockery, John T., "Use of Fuzzy Sets in the Analysis of Military Command", Decision Information, Academic Press, p. 362, 1979.

6. Hersh, H.M. and Caramazza, A., "A Fuzzy Set Approach to Modifiers and Vagueness in Natural Language", Journal of Experimental Psychology, v. 105, no. 3, pp. 254-276, September 1976.

7. Zadeh, L.A., "Quantitative Fuzzy Semantics," Information Sciences, v. 3, no. 2, pp. 159-176, April 1971.

8. Hersh, H.M. and Caramazza, A., "A Fuzzy Set Approach to Modifiers and Vagueness in Natural Language", Journal of Experimental Psychology, v. 105, no. 3, pp. 170-176, September 1976.

9. Zadeh, L.A., "The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems", Fuzzy Sets and Systems, v. 11, no. 3, pp. 199-227, November 1983.

10. NAVSUP Publication 485, Afloat Supply Procedures, Revision 1, Change 1, pp. 6/1-6/235, September 1984.

11. Ibid. pp. 6/153-6/164.

12. Dockery, John T., "Use of Fuzzy Sets in the Analysis of Military Command", Decision Information, Academic Press, p. 364, 1979.

13. Ibid. p. 366.

BIBLIOGRAPHY

Tymshare Inc., REVEAL Reference Manual, 1984

Wang, P.D. and Chang, S.K., Fuzzy Sets, Plenum Press, 1980

Zadeh, L.A., et al, Fuzzy Sets and Their Applications to Cognitive and Decision Processes, Academic Press, Inc., 1975

INITIAL DISTRIBUTION LIST

		No. Copies
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943	2
3.	LCDR Paul Fischbeck Code 55FB Naval Postgraduate School Monterey, CA 93943	1
3.	LT Douglas W Verhagen Navy Fleet Material Support Office Mechanicsburg, PA 17055	3
4.	Professor Norm Lyons Code 54LB Naval Postgraduate School Monterey, CA 93943	1
5.	Navy Ocean Systems Command Code 440 San Diego, CA 92152	1
6.	LCDR David Nickels Navy Strike Warfare Command Naval Air Station, Fallen Fallen, NV 89406	1
7.	Aircraft Weapons Integration Department Code 31 China Weapons Center China Lake, CA 93555	1
8.	Rome Air Development Center Air Force Systems Command Griffis Air Force Base, NY 13441	1
9.	Neal Rowe Code 52RP Naval Postgraduate School Monterey, CA 93943	1
10.	Tymshare, Inc. Attn: Dr. Peter J. Wong Knowledge Engineering Division 20705 Valley Green Drive Cupertino, CA 95014	1
11.	Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
12.	Computer Technology Programs, Code 37 Naval Postgraduate School Monterey, CA 93943	1

213178

Thesis
V413
c.1

Verhagen
Users guide to approximate reasoning and the REVEAL software system.